Operating Instructions

Personal Computer

JR-100U

Panasonic

Before operating this set, please read these instructions completely.

Caution

This personal computer is made of sturdy materials, which provide adequate protection under normal use. There is, however, a limit to the amount of twisting, bending and dropping this unit can withstand. Reasonable care is required to protect the electric parts such as LSIs, ICs, etc. and other components.

Important Don'ts

- Do not use or place this personal computer in areas of high or low temperature, high humidity, direct sunlight, or a dusty atmosphere for a long time. Excessive exposure to these conditions could result in poor performance, damage to the cabinet, or functional failure of the LSI or other components.

  Also, avoid using the personal computer near audio equipment since interference may result.
- Do not shock or expose to water since this will adversely affect performance.
- Do not open the cabinet.

  If the personal computer is defective, please contact the nearest Panasonic Service Center.
- Do not use thinner, benzine or alcohol to clean the personal computer.

  Use a silicone treated cloth, or a cloth dampened with gentle cleaning liquid that will not damage the personal computer.
- Do not use any AC adaptors other than the optional designated Panasonic AC adaptor which is specially designed for use with this personal computer.

CONTENTS

CHAPTER 1 "GENERAL DESCRIPTION OF HARDWARE"

CHAPTER 2 "OPERATION OF PANASONIC PERSONAL COMPUTER JR-100U"

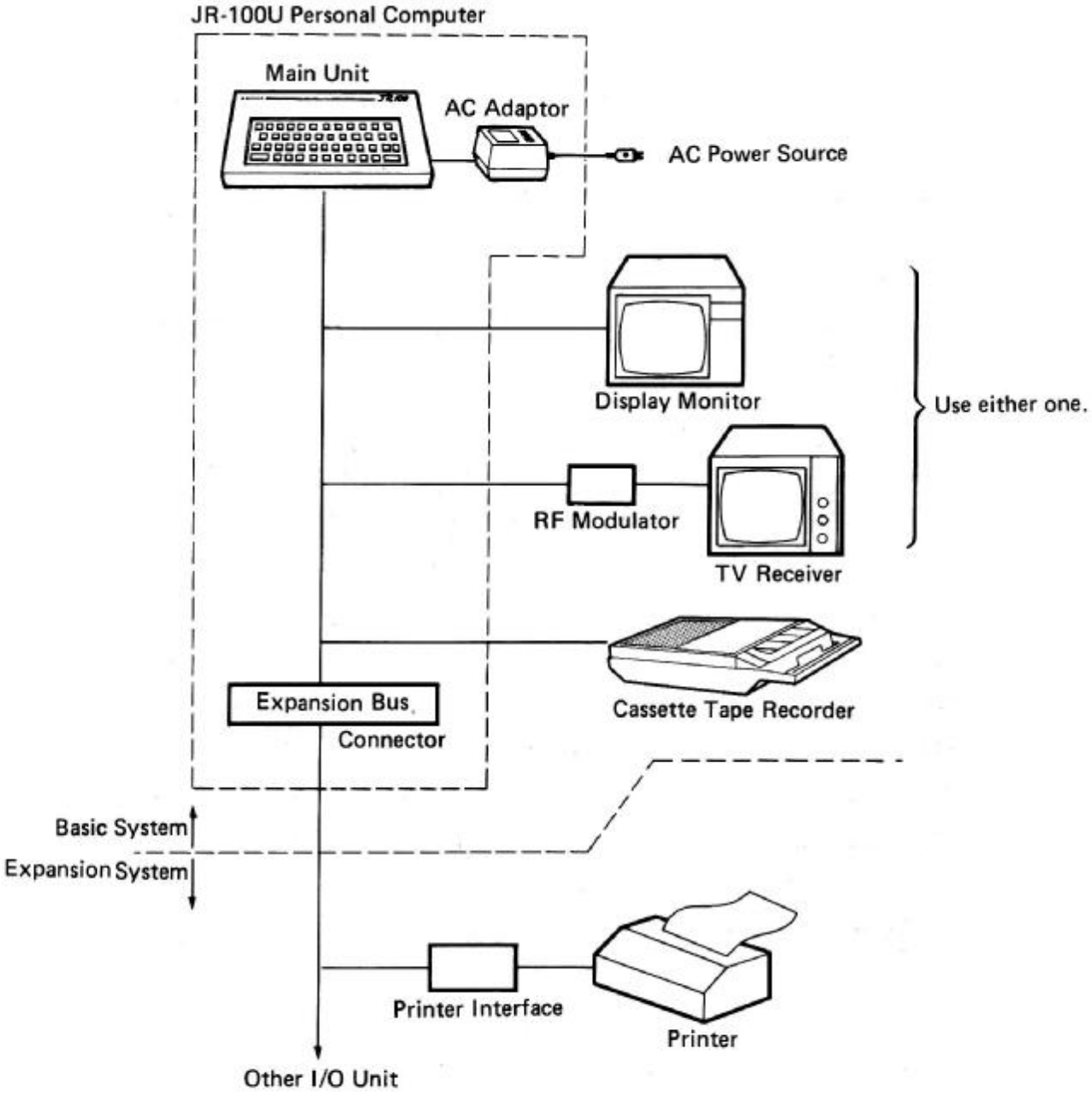
CHAPTER 3 "SPECIFICATIONS OF BASIC"

CHAPTER 1 "GENERAL DESCRIPTION OF HARDWARE"

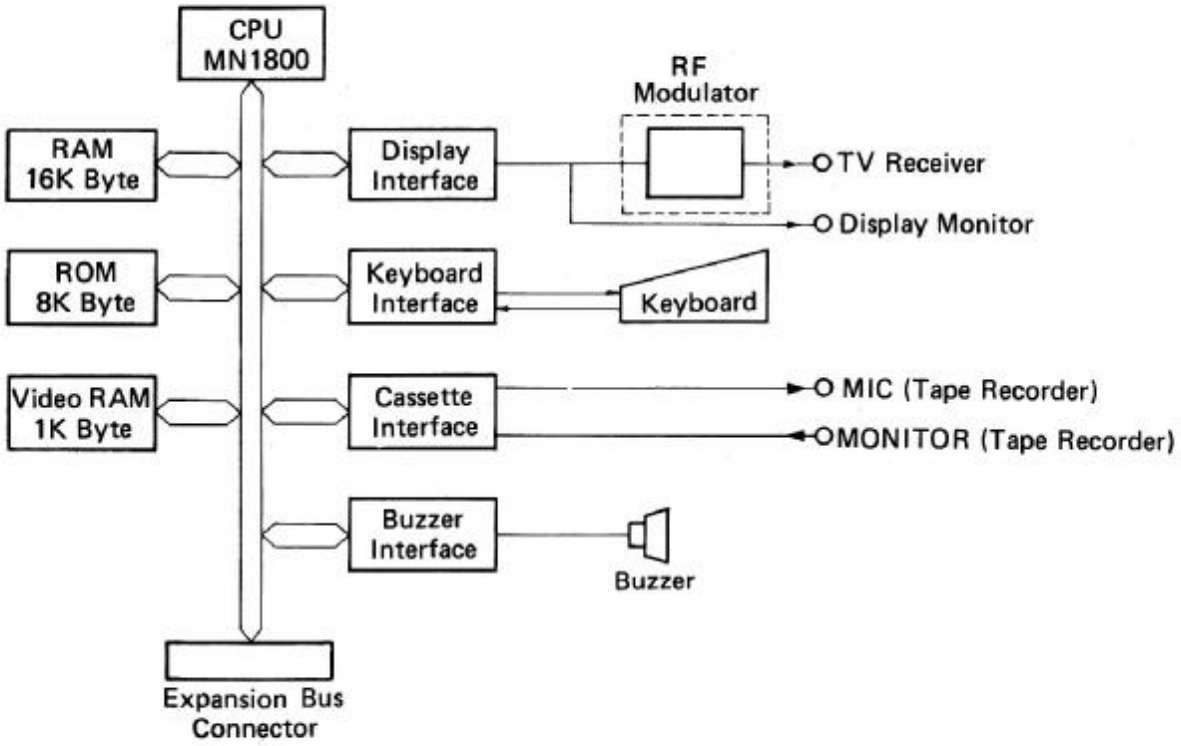
Section 1 – System Configuration and Functional Specifications –

1.1 System Configuration

Panasonic personal computer JR-100U has a basic configuration in which a display monitor (including a family TV monitor) and cassette tape recorder are utilized as the display and the memory, respectively; and various optional configurations are also provided in which various I/O devices including a printer are connected through a printer interface to bus connectors.



A block diagram of the personal computer JR-100U is shown below:



1.2 Functional Specifications

(1) Microprocessor

- Model MN1800 (equivalent to 6802)
- Clock frequency: 890 kHz
- System reset function

(2) Memory

- ROM: 8K bytes
- RAM: 16K bytes
- Video RAM: 1K byte

The capacity of the memory may be optionally increased by 32K bytes through bus connectors.

(3) Keyboard

- System: Software scanning
- Keys: 5-shift key mode with 45 keys, SHIFT key and CTRL key

(4) Display interface

- Screen size: 24 lines x 32 characters with monochrome display monitor
- Characters: 64 characters with 6 x 7 dot matrix
  64 semi-graphic characters with 8 x 8 dot matrix
- Characters and symbols specified by users:
  32 characters with 8 x 8 dot matrix

- Attribute: Inverted display function
- Composite video signal: with 75 ohms, 1 V p-p or with RF flip-flop converter

(5) **Cassette interface**

- System: FSK system 1,200 Hz (space) and 2,400 Hz (mark)
- Baud rate: 600 Bauds

(6) **AC adaptors (JR-A03, JR-A04 and JR-A05, AC 110V, 120V and 220V, respectively.)**

- Input voltage: AC 110 V, 120 V or 220 V ±10%, 50/60 Hz
- Output voltage: DC 17 V, 7.8 V and −8 V
- Power consumption: 12.5 W

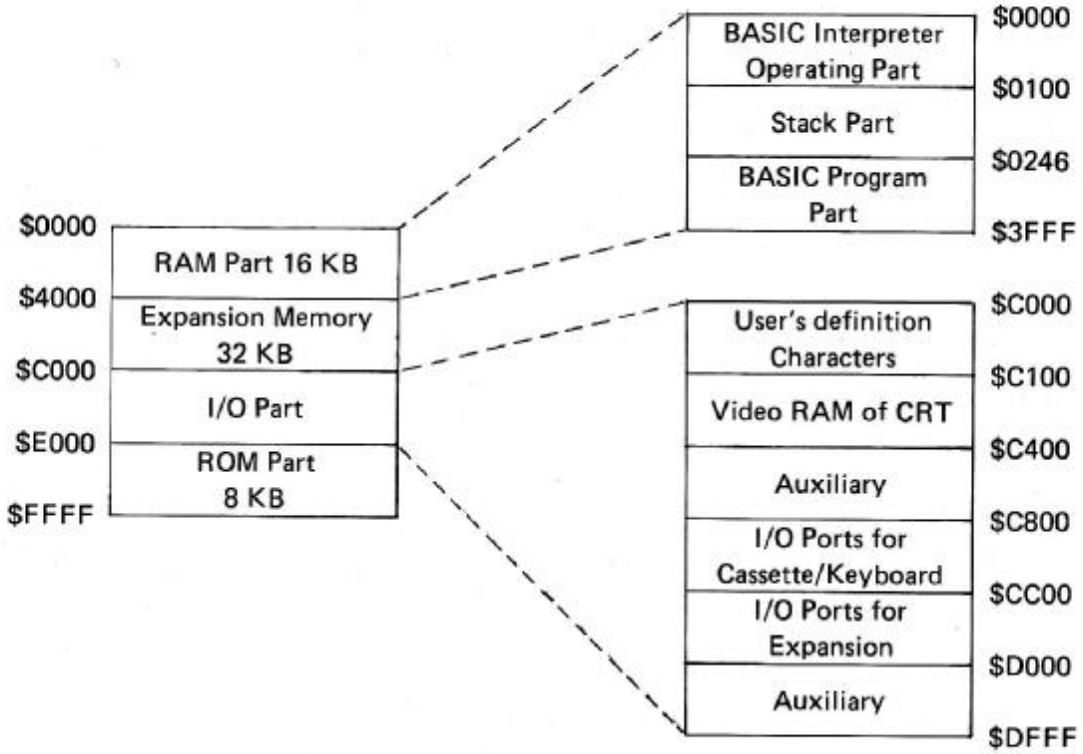
(7) **RF Modulator (JR-R05)**

- For switching TV channel 3 or 4

Section 2 — Memory Mapping —

2.1 Arrangement of Memory

The memories of the personal computer JR-100U are a ROM of 8K bytes for storing control programs of a BASIC interpreter, a screen editor, a character pattern and I/O devices; a RAM of 16K bytes for storing user's programs; and a video RAM of 1K bytes for refreshing a CRT. The configuration of the memory, the cassette magnetic tape, and I/O ports of the keyboard is shown below:



- RAM unit (16 Kbytes)

  This RAM is used for storing BASIC programs and the operating area for the BASIC interpreter. Programs for machine language are stored in an area which does not cor-

responds to the BASIC programs.

- Optional memory unit (32 Kbytes)

  An external memory to be added is mapped in this area through bus connectors.

- I/O Ports (8 Kbytes)

  These I/O ports are used for storing data in or retrieving data from the cassette magnetic tape by keyboard operation with the CRT display.

- ROM unit (8 Kbytes)

  This ROM stores control programs of the BASIC interpreter, the screen editor, the character pattern and the I/O devices.

- Video RAM unit

  The display position on the screen of the CRT display corresponds to a memory address in one-to-one correspondence.

2.2 Application of Machine Language

The following procedure is made when machine language is used with BASIC.

A BASIC program is written in addresses after address $0246. Bits for variables are provided for the BASIC program. The machine language is written after this variable area to address $3FFF.

When the BASIC program area extends to, for example, address $2FFF, the machine language is written in addresses $12 and $13. In the example described above, the following operation is required.

```
POKE $12, $2F
POKE $13, $FF
```

When power is supplied, the data of address $3FFF is automatically written in addresses $12 and $13.

2.3 Access to Video RAM

Data of 1 byte (POKE) is written in a predetermined address of the video RAM to write a character at any position on the CRT screen. The position on the screen corresponds to the predetermined address of the video RAM as follows:

	0	1	——— Digit ———	31
0	C100	C101		C11F
1	C120	C121		C13F
Line			Example: 12 Digits 4 Lines C18C	
22	C3C0	C3C1		C3DF
23	C3E0	C3E1		C3FF

The upper four digits of a character code which is written in the memory differ from that of an ASCII code. A ROM code in a character code table to be shown in the next section is used. For example, when letter "A" is displayed at the 12th digit of the fourth line, the following procedure is required.

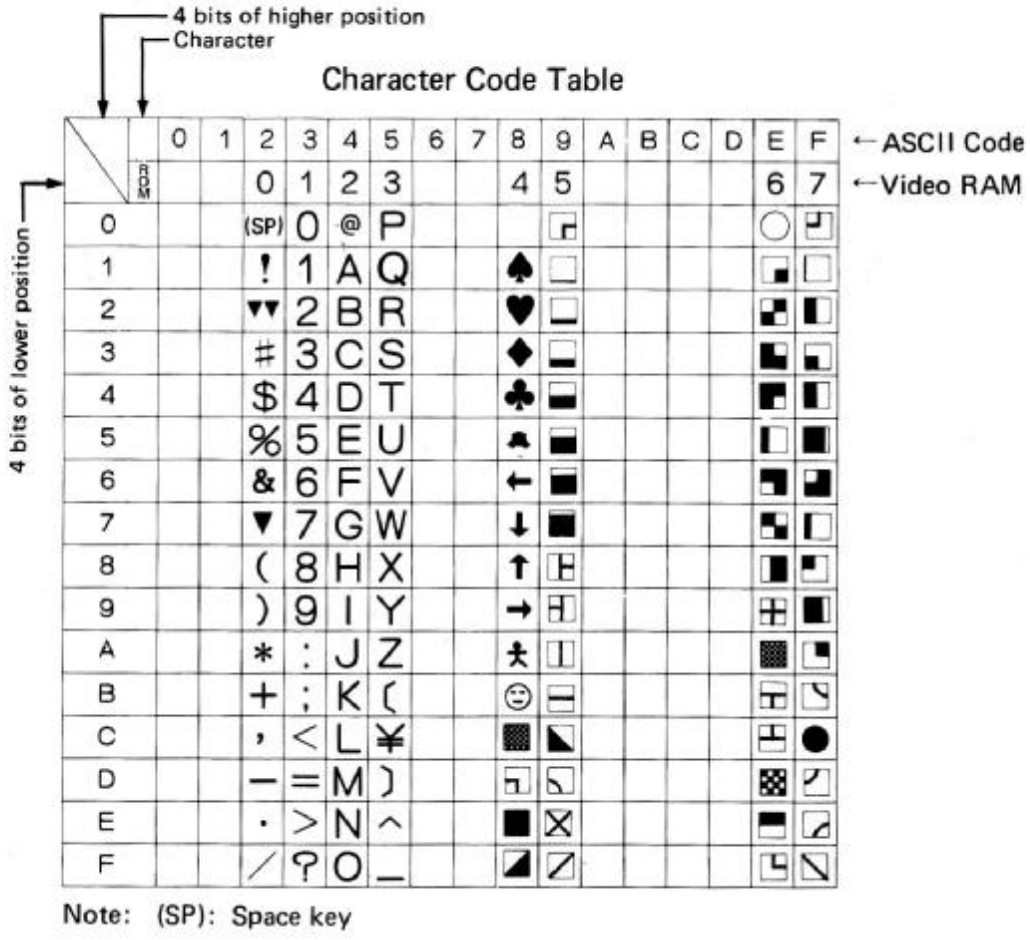
```
POKE $C18C, $21
```

The same results are obtained when the following procedure is performed.

```
10 LOCATE 4, 12
20 PRINT CHR$ ($41)
```

Section 3 — Character Code Table —

One character is written with one byte (8 bits). The character code table is defined as the table in which characters are mapped with 0 to 255 bits ($00 to $FF).



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ROM			0	1	2	3			4	5					6	7
0			(SP)	0	@	P									○	
1			!	1	A	Q			♠							
2			▼▼	2	B	R			♥							
3			#	3	C	S			♦							
4			$	4	D	T			♣							
5			%	5	E	U										
6			&	6	F	V			←							
7			▼	7	G	W			↓							
8			(	8	H	X			↑							
9			)	9	I	Y			→							
A			*	:	J	Z										
B			+	;	K	[			☺							
C			,	<	L	¥										●
D			−	=	M	]										
E			.	>	N	^			■							
F			/	?	O	_										

Note: (SP): Space key

Inverted display is performed for the characters in the character code table. However, an inverted character and a character or symbol specified by the user cannot be simultaneously displayed on the screen.

The pattern of the characters in the character code table cannot be changed since the permanent program for the characters is stored in the ROM. The user may create special

characters or symbols with a desired pattern in units of an 8 x 8 (64 dots) dot matrix as needed.

User's Definition Character Code Table

Upper 4 bits / Lower 4 bits	2		3	
	Designation character	Memory address	Designation character	Memory address
0	(SP)	$C000 – $C007	0	$C080 – $C087
1	!	$C008 – $C00F	1	$C088 – $C08F
2	"	$C010 – $C017	2	$C090 – $C097
3	#	$C018 – $C01F	3	$C098 – $C09F
4	$	$C020 – $C027	4	$C0A0 – $C0A7
5	%	$C028 – $C02F	5	$C0A8 – $C0AF
6	&	$C030 – $C037	6	$C0B0 – $C0B7
7	▼	$C038 – $C03F	7	$C0B8 – $C0BF
8	(	$C040 – $C047	8	$C0C0 – $C0C7
9	)	$C048 – $C04F	9	$C0C8 – $C0CF
A	*	$C050 – $C057	:	$C0D0 – $C0D7
B	+	$C058 – $C05F	;	$C0D8 – $C0DF
C	,	$C060 – $C067	<	$C0E0 – $C0E7
D	–	$C068 – $C06F	=	$C0E8 – $C0EF
E	.	$C070 – $C077	>	$C0F0 – $C0F7
F	/	$C078 – $C07F	?	$C0F8 – $C0FF

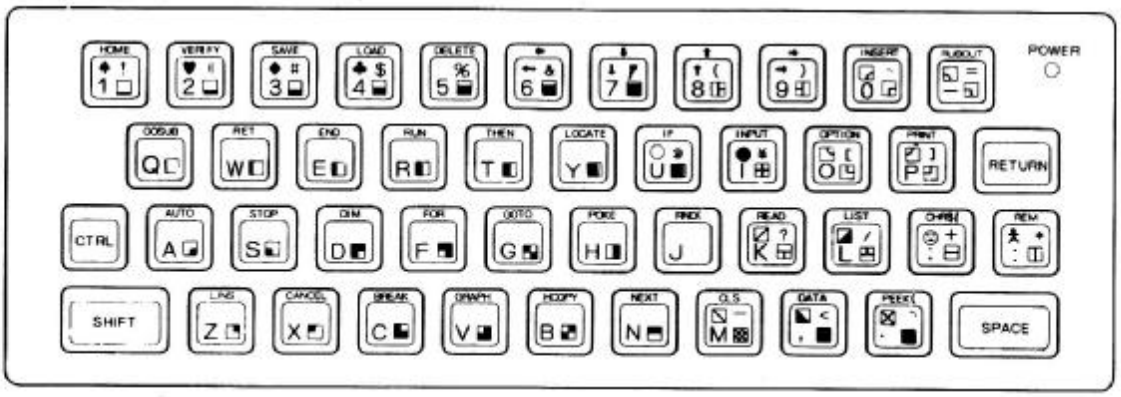
Designation character: a character designated by the user, corresponding to a pattern in place of are given character patten in the program.

Note: (SP): Space key

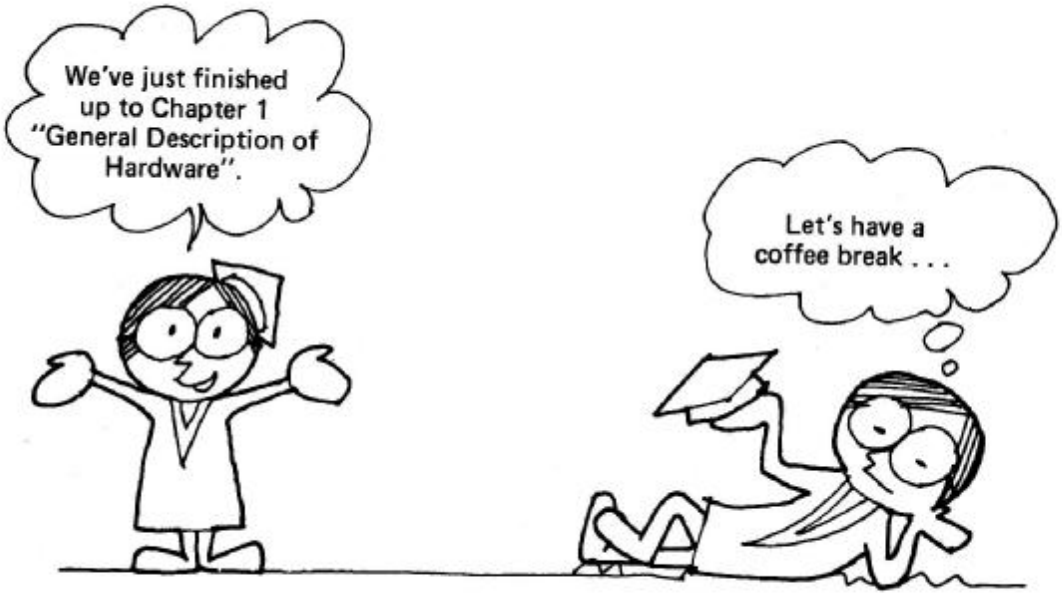
Since the characters or symbols are created by the user, keys which correspond to the created characters or symbols are not present on the keyboard. Each of the keys for special characters or symbols created by the user for practical convenience in order to perform programming. A character pattern specified by the user is written in the corresponding memory address. When this pattern is used in programming, a special character which corresponds to the character pattern specified by the user is used.

(Refer to 4.2 of "BASIC Programming Guide for JR-100U")

Section 4 – Keyboard –

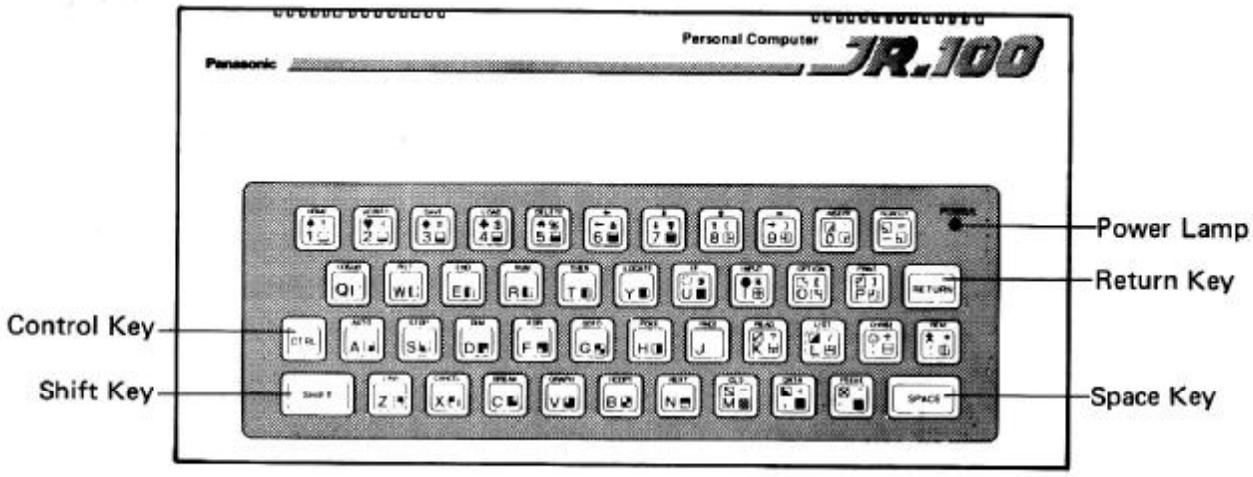


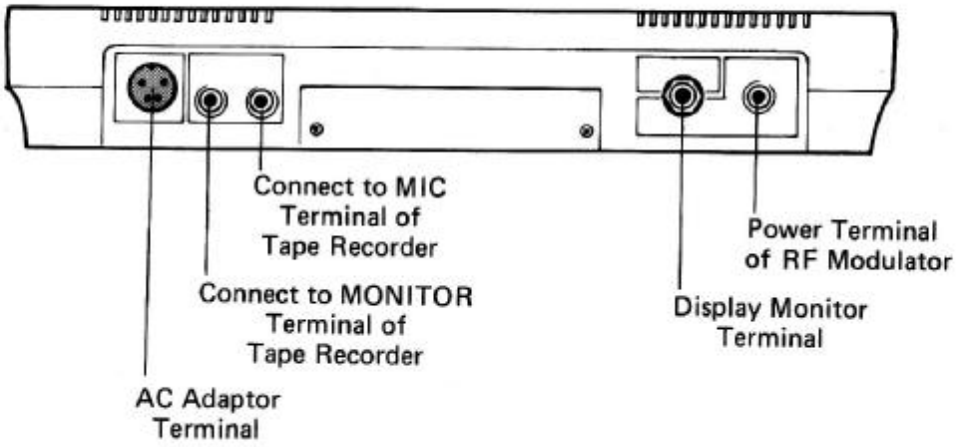
There are 45 keys on the keyboard. Most of the keys can be used in five ways in accordance with the graphic mode, the character mode, the function mode and the **SHIFT** key. Be sure to press the key firmly.



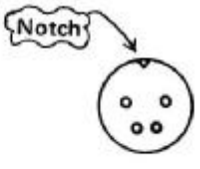
CHAPTER 2 "OPERATION OF PANASONIC PERSONAL COMPUTER JR-100U"

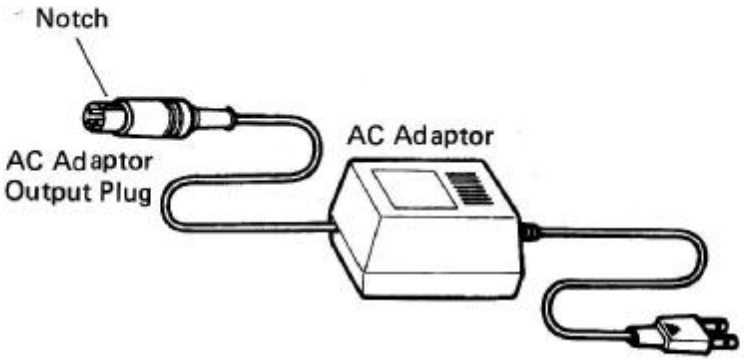
Section 1 — Outer Appearance and Parts Name —





Insert the AC Adaptor output plug into the AC Adaptor terminal properly. Fit the notch on the circumference of the plug with the projection on the inner surface of the terminal.



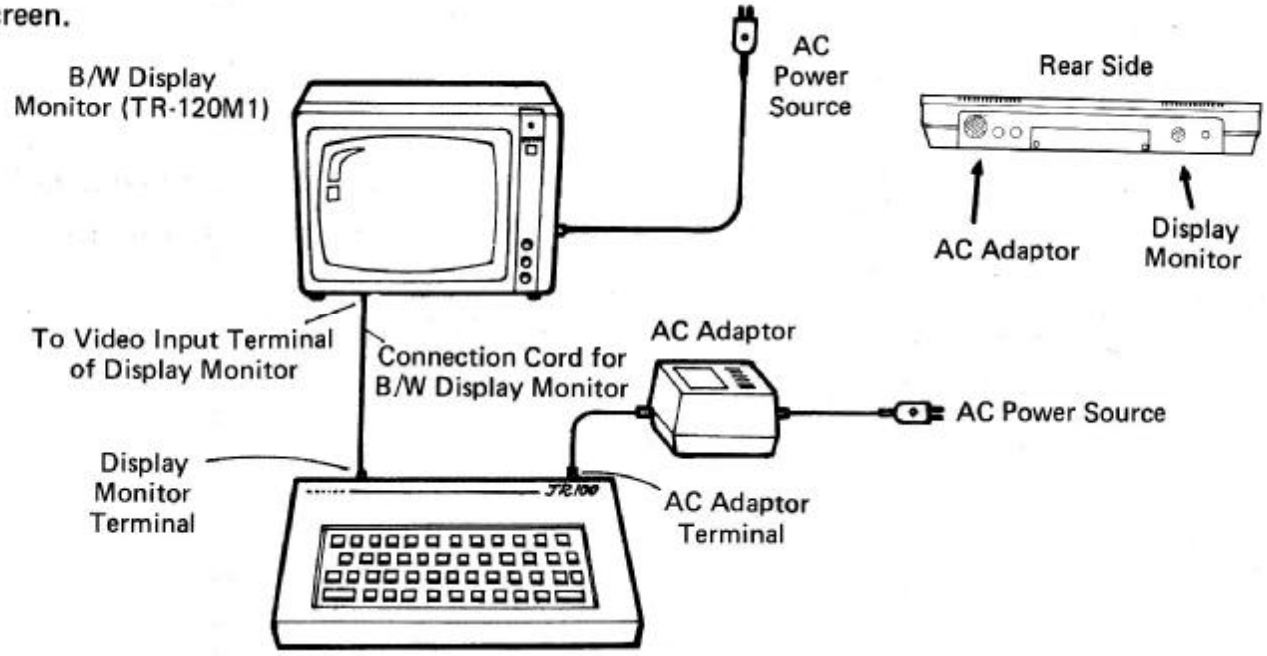


Section 2 — Connections —

2.1 Display

(1) For the display monitor of the JR-100U

Use a monochrome display monitor (Panasonic TR-120M1) to obtain clear images on the screen.

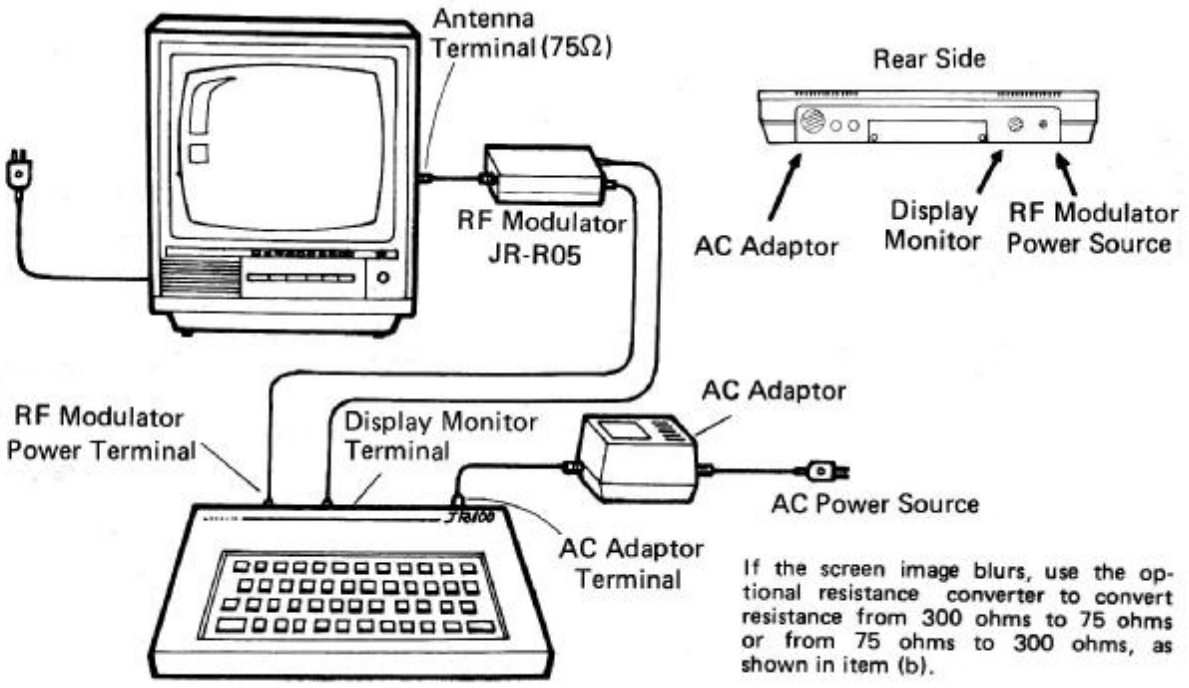


(2) For a family TV

Use the RF Modulator (JR-R05; optional) to switch channel 3 or 4 which does not correspond to a TV station. Set the TV channel to the channel of the RF modulator. Adjust the fine tuning knob to obtain clear images on the screen.

Refer to the Operating Instructions attached to the RF modulator for optimum applications.

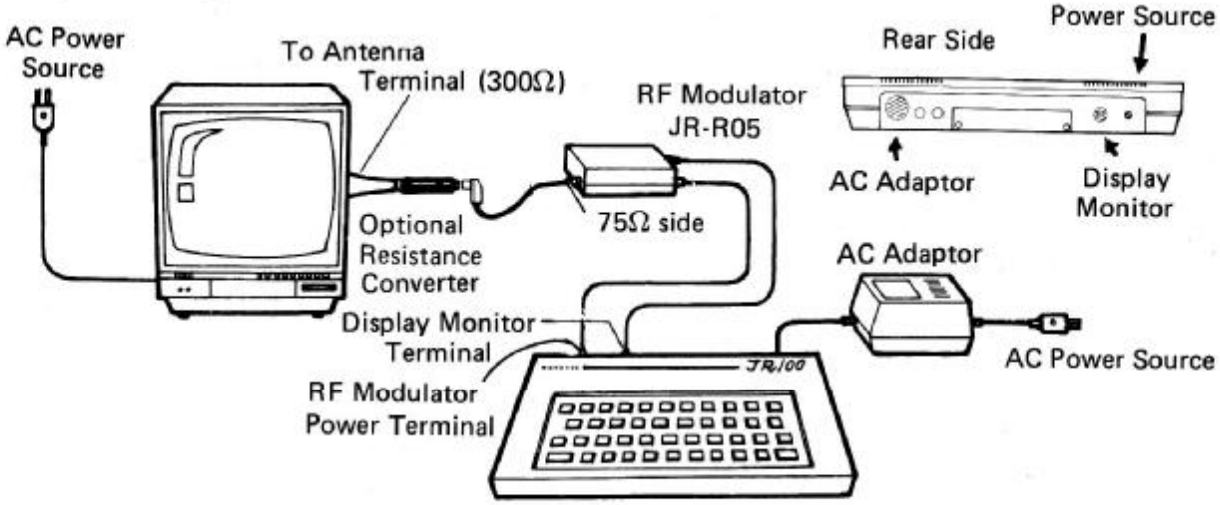
(a) For a coaxial antenna terminal (75 ohms) used for color monitor, etc.



If the screen image blurs, use the optional resistance converter to convert resistance from 300 ohms to 75 ohms or from 75 ohms to 300 ohms, as shown in item (b).

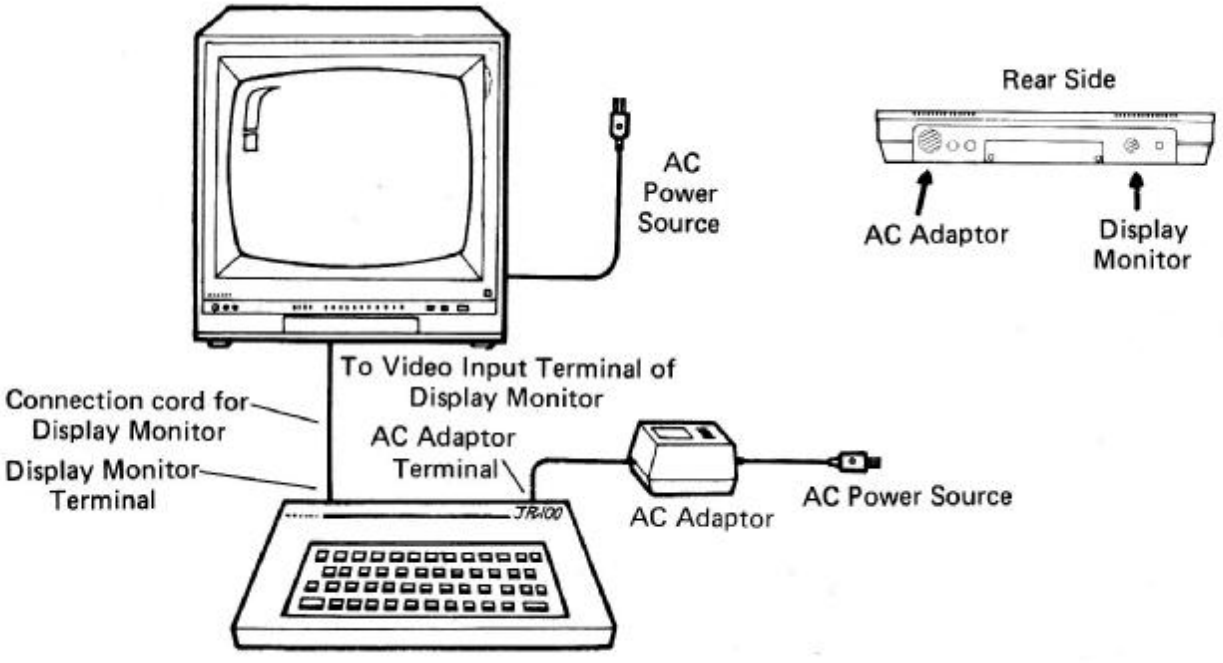
(b) For a balancing antenna terminal (300 ohms) used for a portable TV, etc.

Remove the coaxial connector of the RF output terminal of the RF modulator and attach optional electric resistance converter to convert the resistance from 300 ohms to 75 ohms. Refer to the Operating Instructions of the RF Modulator for optimum applications.



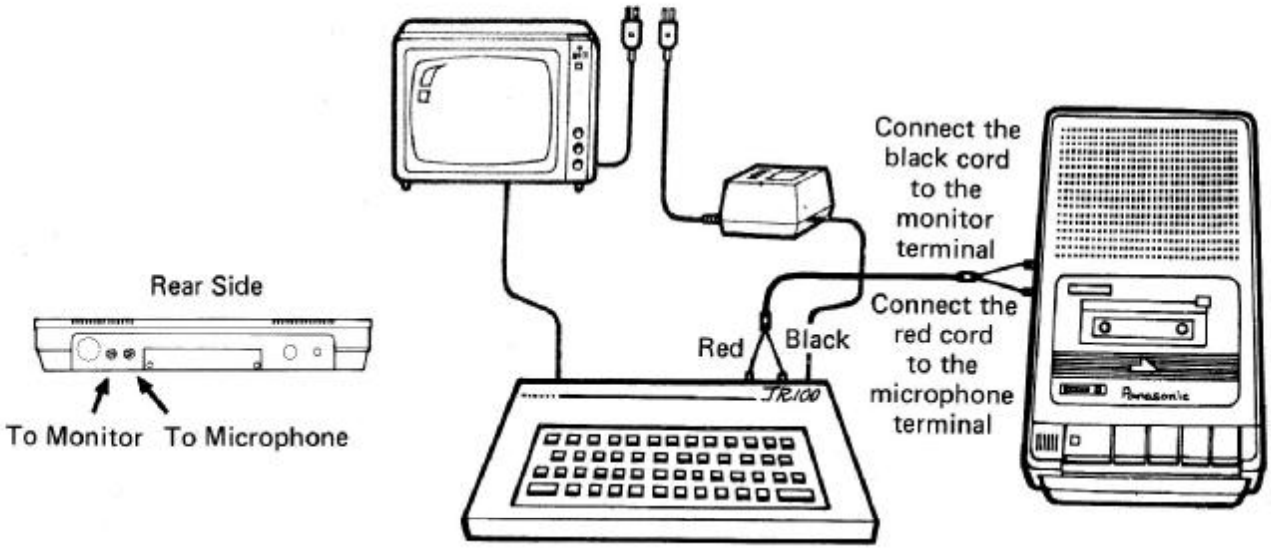
(c) For a TV having video input terminals

Connect the cords in the same manner as in the case in which the special display monitor is used. Further, refer to the Operating Instructions for the TV receiver for optimum applications.



[Note] Details of the characters displayed on the screen may be unclear or blurred when a color monitor is used, depending on the model of the monitor. Perform fine adjustment on the TV receiver side. A monochrome monitor is preferred as the display for the personal computer JR-100U.

2.2 Cassette Tape Recorder



Use the attached recording cable and connect the red plug to the microphone terminal and the black plug to the monitor terminal.

Thus, the main unit is connected to the tape recorder. In some cassette tape recorders, the monitor terminal may be indicated as the external speaker. Refer to the instruction manual of your tape recorder for proper operation.

To Improve the Performance of JR-100U

Remove the cover for the external bus terminals on the back of the main unit of JR-100U and connect expandable equipment (option) to these terminals. Refer to the Operating Instructions of the expandable equipment for further details.

Section 3 – Keyboard Operations –

3.1 Character Mode

- The character mode is the basic mode when power is supplied.
- Press a key to input the character marked (in white) on the left bottom of the key top.
- Simultaneously press the [SHIFT] key and a character key to input the symbol marked (in white) on the upper right of the key top.

3.2 Graphic Mode

- Press the ([V]) key which is marked [GRAPH] while the [CTRL] key is continuously pressed to convert to the graphic mode.
- Press a key to input the symbol marked (in green) on the right bottom of the key top.
- Press the key marked as [GRAPH] while the [CTRL] key is continuously pressed to input a sentence of the BASIC language which is generally displayed on the upper section of the key.

3.3 Function Key Mode

- Press a key while the [CTRL] key is continuously pressed to input a sentence of the BASIC language which is generally displayed on the upper section of the key.

The following operations for the uppermost and lowermost lines are not displayed on the screen.

Control Key Function Key	Function
HOME	To move the cursor to the initial position of the uppermost line
DELETE	To delete the character above the cursor and carry the following lines
RUBOUT	To delete the character immediately before the current cursor position and move the cursor to the position immediately before the current position (Use this function key for a typographical error.)
←↓↑→	To move the cursor one space in the direction shown by the arrows (Use for correcting a program.)
INSERT	To insert a space above the cursor for each key-in operation (Use for inserting a sentence.)

L. INS	To insert a line space (Use for writing a direct command after program correction.)
CANCEL	To cancel an input and move the cursor to the beginning of the line
BREAK	To halt all operations and set the computer in the READY mode
GRAPH	To switch to the graphic mode. Press again to restore the character mode.
HCOPY	To print out the display currently on the screen
SPACE	To input one space
RETURN	To change lines when keyed in at the end of the line, and to input a direct command

3.4 Screen Editor

Programs and sentences displayed containing an error can be corrected on the screen. The functions described above may be utilized for this purpose. For example, "arrows" are used to move the cursor in a desired direction to correct the sentences. Correction on the screen is performed by utilizing the DELETE , INSERT , and RUBOUT keys. When the RETURN key is pressed after correction on the screen, the corresponding program within the memory is corrected in the same manner as displayed on the screen. The RETURN key may be pressed at any time regardless of the cursor position on a given line.

For deleting an arbitrary line, input the line number and press the RETURN key. For inserting a desired sentence between lines, input a line number so that the desired sentence is inserted before the specified line; the line numbers will be arranged in order again. During the period between completion of the STOP sentence (command) and execution of the CONT command, a direct command may be executed, but program correction cannot be performed.

When a long program is reviewed on the screen by executing the LIST command, the images on the screen are rolled up. However, a stationary image may be obtained by pressing a key to review the program step by step.

Since the same type of errors tend to occur, use the FIND function (refer to chapter 3) to list the same type of sentences for convenience in correcting errors.

Section 4 — Operation of Cassette Tape Recorder —

4.1 Connection

Commercially available cassette tape recorders may be used as the display.

Connect the red plug to the MIC terminal and the black plug to the MONITOR terminal to electrically couple the personal computer and the tape recorder. Use a cassette magnetic tape which has a short recording time for avoiding recording trouble.

4.2 SAVE (Store) Program

"SAVE" is an operation for recording a program created by you on the recording magnetic tape.

Reset the counter to "000".

Change a line with the [RETURN] key or the [CTRL] and [Z] keys and follow the procedure as follows by utilizing the [CTRL] key.

```
SAVE "ABC"
```

where "ABC" is the program name. The program name is arbitrarily selected within the range of 15 characters.

Record the program in the tape recorder (press the red recording and play buttons). When a new blank tape is used, wait until the leader of the tape has been wound for about 3 seconds.

Press the [RETURN] key.

On the screen, the following instruction is displayed:

```
PUSH RECORD
```

The following instruction is then displayed on the screen:

```
WAITING ABC
```

In this manner, the SAVE operation is initiated. The word "SAVE" is displayed at the upper right corner of the screen. When the SAVE operation is completed, the following instruction is displayed:

```
READY
```

Press the stop button and rewind the tape to the beginning.

Thus, the SAVE operation is completed. Later on, check with the VERIFY operation if the program is properly recorded.

Increment the counter by at least one to find a nonrecorded section on the magnetic tape when the another program is to be continuously recorded on the same magnetic tape. Mark counter reading and the program name for later application.

Set the tape recorder in the recording mode before pressing the [RETURN] key when the SAVE operation is performed in order to avoid loss of the beginning of the program, because writing is spontaneously initiated upon depression of the [RETURN] key.

4.3 LOAD Program

LOAD is for placing the program into internal storage in the personal computer.

Match the counter number and the corresponding program to be loaded in the computer (the previous program can overlap the desired program).

Change the image on the screen with the [RETURN] key or the [CTRL] and [Z] keys, and follow the procedure with the [CTRL] key as follows:

```
LOAD "ABC"
```

where "ABC" is the program name. Press the [RETURN] key. The following instruction is displayed.

```
PUSH PLAY
SEARCHING
```

The next instruction is then displayed within a short period of time:

```
LOADING ABC
```

The word "LOAD" is displayed at the upper right corner on the screen. When the following instruction is displayed,

```
READY
```

program loading is completed. If the display is:

```
SUM CHECK ERROR !
READY
```

an error has occurred in the loading operation of the program. Turn down the volume level slightly and repeat the LOAD operation of the program to obtain an optimal volume level.

If the magnetic tape is reproduced from the beginning and other programs are recorded

before the desired program on this magnetic tape, the following instruction is displayed:

SKIP ! DEF

Only the desired program is only searched.

Alternatively, when the LOAD operation is only specified without entering the program name as follows:

LOAD

the first program is loaded.

When the LOAD operation of the program is cancelled, perform the BREAK ([CTRL] and [C] key) operation.

4.4 VERIFY

The VERIFY operation is performed in order to check if the program is properly recorded (SAVE) on the magnetic tape. In the same manner as in the LOAD operation, input:

VERIFY "ABC"

When the VERIFY operation is completed, the following instruction is displayed:

READY

However, if the display is:

VERIFY ERROR !

repeat the SAVE operation. During this operation, the program to be recorded again is retained in the memory. If you load another program without recording the program to be recorded, the latter program may be lost.

4.5 MSAVE, MLOAD

The SAVE and LOAD operations in the machine language and data are the same as those in the BASIC language. However, when MSAVE operation is performed, an address in which data is stored must be specified. For example, when data between addresses $3000 and $3100 is stored with the MSAVE operation, input:

MSAVE "ABC", $3000, $3100

where "ABC" is the file name. When data is stored in the MLOAD operation, input:

MLOAD "ABC"

Other steps in the MSAVE and MLOAD operations are the same as in the LOAD and SAVE operations in the BASIC language.

4.6 Notes

The LOAD operation may not be preperly performed depending on the type of tape recorder, even if the fine adjustment of the volume control is performed, as described in

section 4.3. If this occurs, follow the procedure described below:

(1) Insert the plug into the MIC terminal only to perform the SAVE operation. To the contrary, insert the plug into the MONITOR terminal when the LOAD operation is performed. Detach the other plug from the corresponding terminal.

(2) Keep the cassette tape recorder away from the TV monitor.

(3) Adjust the tone if a model with a tone control is used.

Section 5 – Other Operations –

5.1 Control of Buzzer Sound

Buzzer sound may be controlled by inputting data as follows:

Operation	Control
POKE 0, 0	To eliminate buzzer sound for each key-in operation. (except for an error and the BEEP command)
POKE 0, 1	To generate buzzer sound for each key-in operation.
POKE 1, n	To change tone of the buzzer sound. n: $00 – $FF or 0 – 255 and $AA (standard); a high tone is generated when the value of n is small, while a low tone is generated when the value is large.
BEEP	To generate buzzer sound for about 0.5 second.
BEEP 1	To generate buzzer sound.
BEEP 0	To eliminate buzzer sound.

5.2 Other Controls

Operation	Control
RESET	To restore the status at the time of suppling power.
INITP	To initialize the printer.

Symptom	Cause	Remedy
An image is not displayed on the screen.	Power is not supplied.	Check the power source and the power switch.
	Connections are not complete.	Refer to section 2 of chapter 2, connect the cords properly and insert the power plug into the AC power source.
	A proper channel is not set.	Adjust the TV channel to that of RF modulator.
An image is disturbed.	The TV receiver is not adjusted properly.	Adjust tuning, vertical sync. and horizontal sync.
Characters are displayed at random.	The computer is not properly initialized.	Turn off the power switch and turn it on again.
The BREAK function is not initiated.	The computer cannot be controlled (An erroneous program is loaded).	Turn off the power switch, check the program and turn on the power switch again.
The SAVE LOAD function cannot be initiated.	The tape recorder is erroneously operated.	• Refer to section 4 of chapter 2 and perform proper operations. • Adjust the level at the time of loading. • In some tape recorders, insert the plugs into the MIC and MONITOR terminals separately for proper functioning.

CHAPTER 3 "SPECIFICATIONS OF BASIC"

The general specifications of the BASIC language and its commands and sentences will be described below:

Function	To briefly describe the function of a command or a sentence.
Format	To explain the proper format of a command or a sentence.
Summary	To explain a command or a sentence in detail and to clarify the relation of the command or sentence with a corresponding command or sentence.
Example	To summarize basic format of a command or a sentence.
Note	To summarize notes for employing a command or using a sentence.

Proper formats for commands and sentences are described in the format of the BASIC language in accordance with the following descriptive rules.

(1) Input an item represented by letters as it is.

(2) Specify the contents of items with the symbols 〈 〉.
When the contents within the symbols represent a description such as "relation" or "relation n" the relation here means an arithmetic relation.

(3) Omit items with the symbols [ ] since they are optional items.

(4) Be sure to select one element of the contents of an item with the symbols { }, since these are selective.

(5) The symbol | indicates "or".

(6) The symbols ... indicates repetition, within the range of one line of the item immediately before the current item.

(7) Input symbols other than symbols [ ] and 〈 〉.

Section 1 — General Description of Grammar of BASIC —

1.1 Operating Mode

When power is supplied to the Panasonic personal computer JR-100U, the screen displays "READY". This displayed word indicates that the computer is awaiting a command input. Therefore, this status of the computer is the so-called command mode. In the command mode, BASIC commands are executed and BASIC sentences are used as the commands. The latter application is called the direct mode. In the command mode, the cursor is usually displayed at the left end of the line.

When the program is executed with the RUN command, the command mode is automati-

cally converted to the program mode. In this mode, the cursor is not displayed on the screen. The command mode is restored when the program finishing executing a command such as the STOP sentence or the END sentence. Furthermore, the program may be cancelled by the BREAK key operation.

For executing a BASIC sentence in the direct mode, a sentence without a line number is input and the **RETURN** key is pressed to complete the sentence input.

1.2 Line Format and Line Number

The line is a basic unit in the BASIC program as follows:

n [⟨ space ⟩] ⟨ BASIC sentence ⟩ [: ⟨ BASIC sentence ⟩ . . .] **RETURN**

n, which is called the line number, identifies the line and is always placed at the beginning of each line. The range of n is 1 to 32767. When two or more BASIC sentences are written in one line, the sentences are divided by the symbol :. Each line consists of 72 characters at maximum, and the **RETURN** key is always keyed in at the end of each line. In order to emphasize the format of the line, the characteristic features are summarized below:

Line number	1 to 32767
Symbol for dividing BASIC sentences	:
Line length	up to 72 characters

1.3 Constants

Constants are fixed values in a program. The constants consist of value constants and character-string constants.

(1) Value constants: integers between −32767 and + 32767. These constants are described with decimal or hexadecimal notation in the program. When the constants described in hexadecimal notation, the symbol $ is added at the beginning of values.

Ex. Decimal number: −15 235 0 15

Hexadecimal number: $FFF[illegible] $0[illegible] $0 $F

(2) Character-string constants: alphanumeric characters within quotation mar they consist of 32 characters at maximum. A space is counted as a character.

Ex. "ABC␣XYZ" "10 − 100" "♣♦♥♠"

1.4 Variables

Variables are used for temporarily storing data in execution of the progra responding to the stored data is called the variable name. This data is user

value in the program or for recalling the data for a specific purpose.

(1) Types of Variables

The variables are classified into three types in accordance with the types of data.

- Value variable: the range of this variable is −32767 to +32767. The variable consists of data of 2 bytes.
- Character-string variable: the variable consists of data of 32 characters at maximum, and occupies an area of the memory which corresponds to the length of the characters.
- Array variable: this includes a set of data which is accessed by its variable name, although it is included in the value variable.

(2) Names of Variable

- Value variable name: 〈 letter 〉
  〈 letter 〉 [〈 number 〉]
  (example) A C0 X1
- Character variable name: 〈 letter 〉 $
  (example) D$ W$
- Array variable name: 〈 alphabet 〉
  (example) E(5) F(3, 2)

Here 〈 letter 〉 is defined as one of the letters A to Z and 〈 number 〉 is defined as one of the numbers 0 to 9.

1.5 Expressions and Operations

Expressions include constant and variable elements which are connected by operators. The order of operation of the expressions is properly described.

(1) Arithmetic expressions and Operations

Arithmetic expressions include elements such as value variables, value constants, array variables and function values which are combined by arithmetic operations. The operated results are all values. The arithmetic operators are shown below:

Operator	Operation	Example
+	Addition (positive)	A + 5
−	Subtraction (negative)	−B and A − B
*	Multiplication	A * B and C * 2
/	Division	A/B and 10/C

When many operators are included in an arithmetic expression, multiplication and division are performed prior to addition and subtraction. However, when parentheses are present, the elements within the parentheses are first calculated. Various expressions are described in the BASIC language.

〈Expression〉	〈Arithmetic Expression in BASIC Language〉
2x + y	2 * X + Y
x(a + 3)	X * (A + 3)
$x^2 - y^2$	X * X − Y * Y
(a + b) / (a − b)	(A + B) / (A − B)

The function value may also be utilized in the arithmetic calculations.

For adding 1 to the remainder of a/b:

MOD (A, B) + 1

For multiplying 10 times the value obtained by subtracting 1 from the absolute value A:

(ABS (A) − 1) * 10

For producing random numbers in the range between 10 to 29:

RND (20) + 10

For calculating the remainder obtained by dividing the number of characters stored in the character variable C$ by 4:

MOD (LEN (C$), 4)

Refer to section 3 for details on functions.

(2) Character-String Expressions and Operations

The character-string expressions are used for combining character-string variables and character-string function values. The calculated results obtained by combining the character-string expressions with the operator + are character-string data.

$$\left\{\begin{array}{l}\text{Character-string variable}\\ \text{Character-string constant}\\ \text{Character-string function}\end{array}\right\}\left[+\left\{\begin{array}{l}\text{Character-string variable}\\ \text{Character-string constant}\\ \text{Character-string function}\end{array}\right\}\ \ldots\right]$$

In this manner, the character-string expressions are used for combining character-string constants, character-string variables and character-string function values.

If A$ = "A1" and B$ = "B2", and if A$ + B$ is input, then A1B2 is obtained.

If "String" + "DATA" is input, then STRINGDATA is obtained.

If LEFT$ (A$, 1) + RIGHT$ (B$, 1) is input, then A2 is obtained.

If A$ + "␣␣" + B$ is input , then A1␣␣B2 is obtained.

However, if the calculated results are longer than 32 characters, the 33rd character and all following characters are omitted.

(3) Comparison Expression

A comparison expression compares two different value data or character-string data and has a value which is true (1) or false (0). The comparison expression consists of two relations which are combined by a comparison operator.

〈Expression 1〉〈Comparison Operator〉〈Expression 2〉

Expressions 1 and 2 must be the same type of data, regardless of whether they are arithmetic expressions or character-string expressions. The comparison operators are classified into 5 operators as follows:

Comparison Operator	Relation to be Compared	Example
=	equal to	X = Y
>	greater than	A$ > "A"
<	less than	X < 100
>=	greater than or equal to	LEN (A$) >= 5
<=	less than or equal to	X <= Y – 10
<>	not equal to	A$ <> B$

The comparison of the character-string data complies with the following rules.

i) Each of the characters of the respective character-string expressions from the beginning is compared. A space between characters is regarded as a character.

ii) A larger value or a small value of a character is determined by the corresponding larger or smaller value in the character code.

In accordance with the above rules, the comparison of the character-string expressions is judged as follows:

If A$ = "AB", A$ = "AB"
If A$ = "AB", A$ < "AC"
If A$ = "ABC", A$ > "AB"
If A$ = "123", A$ < "124"
If A$ = "12␣", A$ > "12"

In this manner, the left-hand element of the comparion expression is restricted to one character variable.

(4) Logical Expression

Logical Operator	Logical Expression
*	AND
+	OR

The following logical expressions are obtained by combining comparison expressions with a logical operator.

IF (A < B) * (C > D), THEN . . . (AND)

If both comparison expressions (A < B) and (C > D) are true, the operation after "THEN" is performed. If one of the comparison expressions (A < B) and (C > D) is true, or if neither of the comparison expressions (A < B) and (C > D) is true, the operation after "THEN" is omitted and the program advances to the next line.

IF (A < B) + (C > D), THEN . . . (OR)

If both comparison expressions (A < B) and (C > D) are true, or if either of the comparison expressions (A < B) and (C > D) is true, the operation after "THEN" is performed. Otherwise, the operation is omitted and the program advances to the next line.

IF (A < B) + (C > D) = 1, THEN . . . (exclusive OR; XOR)

If only one of the comparison expressions (A < B) and (C > D) is true, the operation after "THEN" is performed. If both comparisons (A < B) and (C > D) are either true or false, the program advances to the next line.

Each element of the logical expression has a logical value of 1 when it is true; the element has a logical value of 0 when it is false. Therefore, the following operation can be performed.

(Example)

The lowest digit of value N including one-digit, two-digit, three-digit and four-digit values is aligned as follows:

```
PRINT SPC ( (N < 10) + (N < 100) + (N < 1000) ) ; N
```

that is,

```
  34
   5
 126
3578
  45
 219
```

1.6 Array Declaration and Element Reference

An array which is a set of data can be accessed by an array variable name. The array variable name declares the name of the array and the number of elements included in this set of data. The array declaration has a format as follows:

〈 Array Variable Name 〉 (〈 Limit 1 〉 [, 〈 Limit 2 〉])

When limit 2 is omitted, the format is called a one-dimensional array; when limits 1 and 2 are specified, the format is called a two-dimensional array. The values of limits 1 and 2 indicates the maximum value, which may be within the range of 1 to 255.

(Example)

i) When array X having 10 elements is declared:

DIM X (10)

ii) When array Y having 5 x 4 = 20 elements is declared:

DIM Y (5, 4)

An elements of the array is specified by an array variable. The format of the array element is as follows:

〈 Array Variable Name 〉 (〈 Expression 1 〉 [, 〈 Expression 2 〉])

wherein the lower limit of expressions 1 and 2 is 1 and the upper limits of the expressions 1 and 2 are, respectively, the values of limits 1 and 2.

The 5th element of the one-dimensional array X is:

X (5)

and the (I * J − 1)th element is expressed as follows:

X (I * J − 1)

Furthermore, the element of the second row and third column is:

Y (2, 3)

and the element on the (I − 1)th column and J row is:

Y (I − 1, J)

1.7 Display Elements and Print Elements

In PRINT and LPRINT sentences, values to be printed and character-string data are specified as display and print elements. These elements include most of the data descriptions which are dealt with in the BASIC language.

$$\left\{\begin{matrix}\text{Display Element}\\ \text{Print Element}\end{matrix}\right\} = \left\{\begin{matrix}\langle\ \text{Expression 1}\ \rangle\\ \langle\ \text{Character-string Variable}\ \rangle\\ \langle\ \text{Character-string Constant}\ \rangle\\ \text{TAB (}\langle\ \text{Expression 2}\ \rangle\text{)}\\ \text{SPC (}\langle\ \text{Expression 3}\ \rangle\text{)}\\ \text{FLD (}\left\{\begin{matrix}0\\ 1\end{matrix}\right\}\text{)}\end{matrix}\right\}$$

Refer to Section 3, Chapter 3 for details on functions.

1.8 User's Definition Characters

One of the characteristic features of Panasonic Personal Computer JR-100U BASIC is the user's definition characters. Various patterns are created by combining normal display and inverted dispaly of the letters, numbers, symbols and semi-graphic characters as shown in the character code table. However, pattern creation by the user's definition characters has a limit.

In order to solve this problem, user's definition characters are created by programming for elaborating character design, For example, a Greek letter, a mathematical symol or a letter having four character regions is displayed. Furthermore, smooth curves and three-dimensional graphics are also drawn.

Creation of the user's definition character is performed with the OPTION sentence (to be described in 2.25 of chapter 3) and the FLD function (to be described in 3.5 of chapter 3).

Section 2 — BASIC Command and Sentences —

2.1 AUTO

Function: To automatically generate a line number prior to a sentence input and to sequentially display the line numbers.

Format: AUTO [〈 Expression 〉]

Summary: The first line number of a specified expression is first displayed, and line numbers in increments of 10 are displayed for each tenth line to assist in input. When input of 〈 Expression 〉 is omitted, the line number begins with 10.

Example:
```
AUTO  100 RETURN
100   PRINT "AUTO" RETURN
110   PRINT "STATEMENT" RETURN
120   RETURN ← Blank line
```

Note: For eliminating automatic display of the line number, press the RETURN key or perform the BREAK key operation.

2.2 BEEP

Function: To control on/off of buzzer sound.

Format: BEEP [0 | 1]

Summary: Buzzer sound is generated in BEEP 1, while buzzer sound is not produced in BEEP 0. When only BEEP is specified, the buzzer sound is produced for about

0.5 second.

Refer to 5.1 of chapter 2 "Control of Buzzer Sound".

Example: BEEP 1 ⎤ Buzzer sound is produced between BEEP 1 and BEEP 0.
BEEP 0 ⎦

2.3 CLEAR

Function: To initialize all the variables.

Format: CLEAR

Summary: Character variables A to Z are set at 0 and other value variable are deleted.

2.4 CLS (Clear Screen)

Function: To delete current displayed contents on the screen and to move the cursor to the home position (upper left corner of the screen).

Format: CLS

Summary: Data displayed on the screen is deleted while the same data stored in the memory is not deleted. This BASIC command is used for changing the contents on the screen at the beginning or midway through the program.

2.5 CONT (Continue)

Function: To restart a program which is interrupted by the STOP sentence.

Format: CONT

Example:

```
〈Program〉                         〈Command〉
                   execution   ←     RUN
     STOP   →  interruption
                                   }Check the program by a direct command.
                   restart     ←     CONT
```

Note: Refer to 2.34 for details on the STOP sentence.

2.6 DATA

Function: To store data in the data pool

Format: DATA 〈Constant〉 [, 〈Constant〉. . .]

Summary: Any DATA sentence of the program may be stored in the memory. The number of constants to be written for one DATA sentence may be arbitrarily selected. The constants are regarded as a series of constants aligned according to the order of the line numbers.

Use the READ sentence when data is to be read out from the data pool. When

the data read out is read out again, the RESTORE sentence (command) must be input for this purpose.

Example: 50 DATA 10, 3
100 DATA ABCD, 0
The above example is the same as the DATA sentence as DATA 10, 5, ABCD, 0.

Note: When the number of data to be read out exceeds the capacity of the data pool, an error (OUT OF DATA ERROR) occurs. The character-string constant need not be put in quotations " "

2.7 DIM (Dimension)

Function: To declare the sets of data to be accessed by the same name and their dimensions.

Format: DIM Array Declaration [, Array Declaration . . .]

Summary: Only array declared by the DIM sentence are only accessed. The array variable name is identified by the array declaration and the element number and its order are determined.

Example: DIM A (10) The number of elements of one-dimensional array variable name A is 10.
DIM A (5, 3) The number of variable elements of two-dimensional array variable name B are 5 x 3 = 15.

Note: The number of the element declared by the DIM sentence is set at 0 at first. The element is referred to by array variable name (expression) or by the array variable name (expression 1, expression 2). The lower limit of the value of the expression is 1 and the upper limit is 255. When the value of the expression is larger than the value specified by the array declaration, this is regarded as an error.

2.8 END

Function: To complete execution of a program.

Format: END

Summary: This command may be omitted when the program is completed at the highest line number. When the END sentence is input, the program mode is interrupted and a new command can be input in the computer.

Example:

```
      ⋮
100 IF A = 0 THEN END
      ⋮
250 END
```

2.9 FIND

Function: To search for a line including a specified character-string and to display the entire part.

Format: FIND 〈Character-string Constant〉

Summary: Since a particular character-string can be specified, this command may be conveniently used for selective display of a line in debugging.

Example: For displaying all the lines including character-string PRINT, input

```
FIND "PRINT"
```

2.10 FOR . . . NEXT

Function: To perform a sentence between FOR and NEXT sentences as many times as specified.

Format: FOR 〈Value Variable〉=〈Expression 1〉 TO 〈Expression 2〉 [STEP 〈Expression 3〉]

. . .

NEXT [〈Value Variable〉]

Summary: Repetition of the FOR . . . NEXT sentence is a controlled loop. Therefore, within the loop, an area which is surrounded by the FOR sentence and the NEXT sentence is repeatedly performed if the value variables of the FOR and NEXT sentences coincide.

The value variable is called the loop control variable within the loop. Some FOR sentences may have a value of 1. When the NEXT sentence (command) is executed, the value of expression 3 is added to the loop control variable and the results are compared with the value of expression 2. If the loop control variable is less than or equal to the value of expression 2, the program advances to the sentence following to the FOR sentence. However, if the loop control variable is greater than the value of expression 2, the sentence following the NEXT sentence (command) is executed.

A multiple loop is performed for nesting the FOR . . . NEXT loops. However, be careful not to cross the FOR . . . NEXT loops.

Example:

```
10 FOR X = 1 TO 10 STEP 2
20 PRINT X
30 NEXT X
```

A loop is repeated five times in which every time X takes a different odd number 1, 3, 5, 7 or 9.

```
10 FOR X=1 TO 9
20 FOR Y=1 TO 9
30 PRINT X;"X";Y;"=";X*Y;"  ";
40 NEST Y
50 PRINT
60 NEXT X
```

Double loop. A table including 81 values is displayed as products of 1 x 1, ... , 9 x 9

Note: Be careful to note that a loop is performed once when expression 1 is larger than expression 2. The multiple loop may be performed ten times. STEP 1 may be omitted.

2.11 GOSUB . . . RETURN/RET

Function: To access a subroutine and to advance to a next sentence.

Format: GOSUB 〈 Expression 〉

The format may be abbreviated in RETURN ← RET.

Summary: A sentence whose line number is the expression value is accessed for the beginning of the subroutine and the RETURN sentence is input at the end of the subroutine. The next sentence is then initiated in the GOSUB command. The subroutine may be accessed in the program as many times as needed.

A subroutine may be placed in any one of the main programs. The RETURN sentence must be input at the end of the subroutine.

Example: For accessing a subroutine beginning from line number 500:

```
100 GOSUB 500
.....
500 REM SUBROUTINE
.....
550 LET RETURN
```

Note: When a subroutine is accessed in the subroutine, be careful not to exceed an accessing depth of more than 15. When the value of the expression does not have a corresponding line to access, an error occurs.

2.12 GOTO

Function: To jump to a sentence whose line number is indicated by the value of the expression.

Format: GOTO 〈 Expression 〉

Summary: When this sentence (command) is executed, the program sequence is changed and the sentence whose line number is indicated by the value of the expression

is accessed.

Example:
```
100 GOTO 150          Jump to line number 150
.....
150 K = 220
.....
200 GOTO K            Jump to line number 220
.....
220 GOTO 220          Jump to line number 220 (Infinite loop)
```

Note: When the expression value indicates a line number which does not exist in the program, an error occurs. Use the BREAK key operation to escape from the infinite loop.

2.13 HCOPY (Hard Copy)

Function: To print a currently displayed image on the screen.

Format: HCOPY

Summary: When continuously changing images on the screen are to be printed, input the HCOPY sentence for your convenience.

Note: Use the BREAK key operation to interrupt printing. If the HCOPY function mode key is used, the printing operation is performed without displaying the printed contents on the screen.

2.14 IF . . . THEN

Function: To change the program sequence in accordance with the conditions specified by a comparison expression.

Format: IF 〈 Expression 1 〉 Comparison Operator 〈 Expression 2 〉 [THEN]

Summary: If the compared results are true, the sentence after "THEN" is executed. If the results are false, the next sentence is executed.

Example:
```
IF A <= 0 THEN GOTO 100
```
(which indicates that if A is zero, execute GOTO 100)
```
IF BS = "STRING" THEN S = 1 : GOTO 50
```
(which indicates that if BS is equal to character-string STRING, first execute S = 1 and jump to line number 50.)

Note: If the expressions to be compared are not the same type (value or character-string), an error occurs.

2.15 INPUT

Function: To input data with the keyboard and to substitute the data for a variable.

Format: INPUT [⟨ Prompt Sentence ⟩ ,] ⟨ Variable ⟩ [, ⟨ Variable . . .]

where ⟨ Prompt Sentence ⟩ is " ⟨ Character-string ⟩ "

Summary: When this sentence (command) is input, a question mark ? is displayed on the screen to request the data to be input. If the prompt sentence is present, this sentence is displayed before the question mark ? so that data to be input may be easily determined. The number, order and type of data to be input after the question mark ? must coincide with attributes of the variables of the input sentence. Data to be input are divided by commas to be distinguished.

Example:
```
10 INPUT A, B, C$
20 INPUT A, B, C$
RUN
? 5, -10, INPUT RETURN ← Depress RETURN at end of the input data.
5   -10   INPUT
100 INPUT "DATA", X
RUN
DATA ? 100 RETURN
```

Note: Adjust the number of variables so that the number of characters for data to be input with one sentence is within 72 characters. When an input is requested and only the RETURN key is depressed, the question mark ? is displayed on the screen and the program awaits data to be input.

2.16 LET

Function: To evaluate an expression and to replace a variable with the calculated results.

Format: [LET] ⟨ Variable Name ⟩ = { ⟨ Arithmetic Expression ⟩ | ⟨ Character-String Expression ⟩ }

Summary: The key word LET may be omitted. Substitution is properly performed only if the right hand element and the left hand element coincide in type.

Example:
```
10 LET I = 5                ← Substitute 5 for I
20 LET J = I + 3            ← Substitute 8 (result of 5 + 3) for J
30 LET K = I * I + J        ← Substitute 33 (result of 5 x 5 + 8) for K
40 LET A$ = "LET␣"
50 LET B$ = "STATEMENT"
60 LET C$ = A$ + B$         ← Substitute LET STATEMENT for C$
   ↑
   └ All LETs can be omitted.
```

2.17 LIST

Function: To display all or part of a program stored in the memory.

Format: LIST [〈 Line Number 〉 [, 〈 Line Number 2 〉]]

Summary: When line numbers are not specified, the entire program is displayed. When line number 1 is specified, the corresponding line is displayed. Further, when line number 1 and 2 are specified only, the first and second lines are displayed.

The line numbers are displayed beginning with the smallest numbers. When the specified line may not be entirely displayed on the screen, the currently displayed image is rolled up to display the rest of the line. Continuously press an arbitrary key (excluding the BREAK key) to obtain a tationary image in this command.

Example:

LIST	Entire program is displayed.
LIST 100	Only line number 100 is displayed.
LIST 100, 200	Contents between line numbers 100 and 200 are displayed.
LIST 100, 0	Contents after the line number 100 are displayed.

2.18 LLIST

Function: To print all or part of the program stored in the memory.

Format: LLIST [〈 Line Number 1 〉 [, 〈 Line Number 2 〉]]

Summary: The line numbers are specified in the same manner as in the LIST command.

Note: Use the BREAK key to interrupt printing.

2.19 LOAD

Function: To retrieve a BASIC program from the cassette tape and to store it in the memory.

Format: LOAD [" 〈 File Name 〉 "]

where 〈 File Name 〉 is a character-string of up to 15 characters.

Summary: When the LOAD command is executed, the instruction PUSH PLAY is displayed. Set the cassette tape recorder in the reproducing mode. The status LOADING is displayed, and the file name is then displayed.

When the file name is omitted, the fist file searched is input after the command input. When a file name is specified, searching is performed until the corresponding file is found. Even if other files are found, the instruction SKIP 〈 File Name 〉 is displayed so that proper searching is assured.

When input of the specified file is completed, READY is displayed on the screen so that turn the tape recorder may be turned off.

Example: LOAD "GAME"

(which indicates that file name GAME is input.)

LOAD

(which indicates that the first file searched from the current position of the cassette magnetic tape is input.)

Note: When a reading error occurs during input of data, the instruction SUM CHECK ERROR is displayed on the screen. Check its cause by referring to section 6, chapter 2. Use the BREAK key to interrupt the data input.

2.20 LOCATE

Function: To move the cursor to an arbitrary position on the screen.

Format: LOCATE 〈 Expression 1 〉, 〈 Expression 2 〉

Summary: When the upper left corner is defined as the origin (0, 0) and the lower right corner is defined as (23, 31), the cursor may be moved to an arbitrary position within this range.

Example: Display symbol at 13th digit in 5th line .

```
LOCATE 5, 13 : PRINT "  "
```

Display I at 8th digit from 2nd line to 10 th line

```
10 FOR I = 2 TO 10
20 LOCATE I, 8
30 PRINT "|";        ← | : □ of Graphic mode
40 NEXT I
```

Note: Specify the range of expressions 1 and 2 as follows:

$0 \leqq$ Expression 1 $\leqq 23$ and $0 \leqq$ Expression 2 $\leqq 31$

2.21 LPRINT

Function: To print value data or character-string data at the printer.

Format: LPRINT 〈 Print Element 〉[〈 Division Symbol 〉〈 Print Element 〉. . .]

Summary: This command is the same as the PRINT command except that the results are printed out at the printer.

2.22 MLOAD

Function: To retrieve the machine language recorded on the cassette magnetic tape and to stored it in the memory.

Format: MLOAD [" 〈 File Name 〉 "]

Summary: This command is the same as the LOAD command except that the retrieved language is machine language, while BASIC is retrieved in the case of the LOAD command.

2.23 MSAVE

Function: To put a file name for a machine language program stored in the memory and to record it on the magnetic tape.

Format: MSAVE ["〈File Name〉"] ,〈Start Address〉,〈End Address〉

Summary: This command is the same as the SAVE command, except that the machine language is recorded on the magnetic tape, while the BASIC language is recorded on the magnetic tape, while the BASIC language is recorded on the magnetic tape in the SAVE command. An address can be accessed by a decimal number or a hexadecimal number with $.

2.24 NEW

Function: To delete all the BASIC programs stored in the memory.

Format: NEW

Summary: When a new program is input, the previously stored programs are cleared.

2.25 OPTION

Function: To select the screen display mode or the overflow processing mode.

Format: OPTION { CMODE0 | CMODE1 } | { OVF0 | OVF1 }

Summary: CMODE0: To select the inverted character mode. All the characters in the character code table and their inverted characters are displayed.

CMODE1: To select a user's definition character mode. All the characters in the character code table and the user's definition character code table are displayed.

OVF0: To select the overflow detecting mode which is initiated immediately after overflow occurs.

OVF1: To select the overflow nondetecting mode to continue the program sequence even though overflow occurs.

When a displayed character is specified, use a character in the character code table. However, when an inverted character or a user's definition character is input, the FLD function is utilized to specify either type of character in the printed elements. (Refer to section 3.5 for details on the FLD function.)

Example: For specifying the overflow nondetecting mode:

```
OPTION OVF1
```

For specifying the user's definition character mode:

```
OPTION CMODE1
```

Note: The inverted character mode and the overflow detecting mode are specified

when the OPTION is omitted.

2.26 PICK

Function: To replace, in decimal notation, a variable by the character code of a key which is pressed when this PICK command is performed.

Format: PICK 〈 Value Variable 〉

Summary: If data is not input, input sentences are generally not executed. However, the PICK sentence is executed regardless of the presence or absence of input data. When the PICK sentence is executed, the character code of the key which is pressed is substituted for the value variable. When the key is not pressed, the value of 0 is substituted.

Example: For waiting until a key is pressed for the execution of the PICK sentence:

```
10 PICK K
20 IF K = 0 THEN GOTO 10
```

2.27 POKE

Function: To write data of 1 byte in a specified address of the memory.

Format: POKE 〈 Expression 1 〉, 〈 Expression 2 〉

Summary: The lower byte of expression 2 is written by defining the value of expression 1 as the memory address. In other words, data to be written in the memory must be in the range of 0 to 255 bytes.

The opposite function of the POKE function is the PEEK function (refer to section 3.12). The POKE and PEEK commands are directly written in the memory, and are conveniently used for data exchange with the machine language.

Example: For writing data of 5 in address $1000 of the memory:

```
POKE $1000, 5
```

For writing the values of variable X, which are divided into upper and lower bytes, into addresses 2000 and 2001.

```
POKE 2000, X/256
POKE 2001, X
```

Note: Be careful not to destroy data of the memory area which is used for the BASIC language; refer to section 2, chapter 1 on the memory arrangement.

2.28 PRINT

Function: To display value and character-string data on the display screen.

Format: PRINT 〈 Display Element 〉 [〈 Division Signal 〉 〈 Display Element 〉 . . .]

Summary: When the display element is omitted, only the lines are changed. Refer to subsection 1.7, chapter 3 on the display element.

- Display Position

  The display begins from the cursor position. One line on the screen is divided into four lengths (each corresponding to 8 digits). If division signal is a comma (,), the display element is displayed at the beginning of the next display length. If the division signal is semicolon (;), the display element is displayed immediately after the displayed data.

- Display Continuation

  When a line is completely displayed within the range of 72 characters, automatic line return is performed and data input is continued. When the last element of the PRINT sentence is the display element, the line is returned and the display is completed. If the last element of the line is a comma, the first element of the next display element of the PRINT sentence is displayed in the first position of the subsequent divided length. Furthermore, when the last element of the PRINT sentence is a semicolon, the first display element of the next PRINT sentence is displayed immediately after the end of the current PRINT sentence.

Example:

⟨ Program ⟩	⟨ Display ⟩
PRINT	Line feed
PRINT "ABC" ; "DEF"	ABCDEF
PRINT "ABC" , "DEF"	ABC␣␣␣␣␣␣ DEF

```
10 A = 123
20 PRINT "A =" ; A          A = 123
```

```
10 FOR I = $41 TO $46
20 PRINT CHR$( I );         ABCDEF
30 NEXT I
```

⟨ Display ⟩

A ␣␣␣␣␣␣␣ B ␣␣␣␣␣␣␣ C ␣␣␣␣␣␣␣ D ␣␣␣␣␣␣␣
E ␣␣␣␣␣␣␣ F ␣␣␣␣␣␣␣

⟨ Program ⟩	⟨ Display ⟩
PRINT ▼ ␣ ! " # $ ▼	␣ ! " # $

Use a mark (▼) in place of a mark (") to display the mark (") in the PRINT sentence.

Note: Use a semicolon (;) or a comma (,) for spacing sentence separation as well as SPC, TAB, be FLD functions to be described later for better display.

2.29 READ

Function: To read out data from data pool and substitute it in a variable.

Format: READ 〈Variable〉[,〈Variable〉. . .]

Summary: When the READ sentence is input, data re sequentially read out from the data pool created by the DATA sentence. The readout data are sequentially substituted in variables. Even if the READ sentence is divided into segments, data immediately before the current READ sentence is read out first.

When the readout data is read out again, use the RESTORE sentence (refer to subsection 2.31).

Example:

```
10 DATA 1, 2, 3
20 DATA "DATA", "STATEMENT"
30 READ A, B                ← Equivalent to A = 1 and B = 2
40 READ C, D$               ← Equivalent to C = 3, D$ = "DATA"
.....
100 RESTORE                 ← Return the position of readout data to
                              starting position of data pool.
110 READ I, J, K            ← Equivalent to I = 1, J = 2 and K = 3


10 DIM X (10)
20 FOR J = 1 TO 10
30 READ X (J)               ← Substitute 1 to 10 for X (1) to X (10)
40 NEXT J
50 DATA 1, 2, 3, 4, 5
60 DATA 6, 7, 8, 9, 10
```

Note: All the data need not be read out from the data pool. If the variable of the READ sentence and the type of data to be read out do not coincide, or if many data which exceed the capacity of the data pool are read out, an error occurs.

2.30 REM

Function: To describe remarks of a program.

Format: REM [〈Character-String〉]

Summary: The REM sentences has no function for performing the program so that no influence is given for execution of the program. The REM sentence is used for easy operation of the program as the reference.

Note: If a sentence is added to the REM sentence by dividing them by a colon, this input is neglected.

2.31 RESTORE

Function: To map the data readout position from the data pool with the beginning of the data pool or with the beginning of a arbitrary DATA sentence.

Format: RESTORE [〈 Expression 〉]

Summary: When the expression is omitted, data be read out is mapped from the beginning of the data pool. However, if an expression is specified, the data readout position is mapped from the beginning of the data whose line number is the value of the expression.

Example:

```
10 READ A, B          ← Substitute A = 1 and B = 2
20 RESTORE            ← Restore the start position of the data pool
30 READ I, J          ← Substitute I = 1 and J = 2
40 RESTORE            ← Restore line number 70
50 READ X, Y          ← Substitute X = –1 and Y = –2
60 DATA 1, 2, 3, 4,
70 DATA –1, –2, –3, –4
```

Note: The RESTORE sentence may be repeated. If a sentence whose line number is specified by the expression is not a DATA sentence, the next DATA sentence after the specified line number is searched. If no DATA sentence is present, an error occurs.

2.32 RUN

Function: To begin the program with the sentence having the smallest line number.

Format: RUN

Summary: With this command, the computer is set in the program mode. In this mode, the cursor is not displayed except for executing the INPUT sentence. When the program mode is converted to the command mode, the program is interrupted by the STOP sentence, completed by the END sentence, or canceled by the BREAK key.

Note: When the program is to be operated from the middle, use the GOTO command.

2.33 SAVE

Function: To record a BASIC program with a file name stored in the memory.

Format: SAVE [" 〈 File Name 〉 "]

Summary: Set the cassette tape recorder in the recording mode before inputting this command. When the SAVE command is input, WRITING and the file name are displayed on the screen. When writing is completed, READY is displayed and the command mode is restored. Turn off the cassette tape recorder. The file name may be omitted.

Example: SAVE "GAME1" (which indicates that file name GAME 1 is assigned to a program and this program is recorded on the cassette magnetic tape.)

Note: When the BREAK key operation is performed, the program is restored to the command mode, and the data which have been stored may be lost.

Reading is performed by deleting the data which have been recorded in order to avoid overlapping the data with other data. Especially, when a program which is input in the LOAD command is edited and is written in the same location, the length of the program may be increased so that the gap between the files is sufficient.

2.34 STOP

Function: To interrupt execution of a program and to restore the command mode.

Format: STOP [〈 Display Element 〉 [〈 Division Symbol 〉 〈 Display Element 〉 . . .]]

Summary: The STOP sentence is used for temporarily interrupting the program. Use the STOP sentence as often as needed at anywhere in the program.

When a character-string is input, this character-string is displayed as a unit. Since the program is temporarily interrupted, the value stored in a variable may be checked by the PRINT command in the direct mode, or the program sequence can be changed by the LET command.

For restoring the current program, use the CONT command (refer to section 2.5).

Example:

〈 Program 〉	〈 Command 〉	
10 A = 5	← RUN	
....		
50 STOP	→ PRINT A	← Display A
	A = 3	← Change A to 3
....	← CONT	← Continue

2.35 VERIFY

Function: To verify whether or not a program is recorded on a cassette magnetic tape in the SAVE or MSAVE command.

Format: VERIFY [" 〈 File Name 〉 "]

Summary: The function reads out the program recorded on the magnetic tape and compares the recorded program on the magnetic tape with the program stored in the memory. If the instruction READY is displayed, the recorded program is verified. If the error instruction VERIFY ERROR is displayed, repeat the SAVE command.

The file name may be omitted.

Example: VERIFY "GAME1" (which indicates that the file with file name GAME 1 is read out and compared with the corresponding contents in the memory).

Section 3 – BASIC Functions –

BASIC includes 20 intrinsic functions. These functions can be freely accessed in a program. The value of a value function is given by a number and that of a character-string function is given by a character-string.

3.1 ABS (Absolute Value)

Function: To give the absolute value of an expression.

Format: ABS (〈 Expression 〉)

Example: For obtaining | (A – B)/C |

ABS ((A – B) / C)

3.2 ASC (ASCII)

Function. To give, in decimal notation, the ASCII code of the first character of the character-string data.

Format: ASC (〈 Character-String Expression 〉)

Example:
```
10 A$ = "0123"
20 PRINT ASC (A$)    ← Display 48 (Decimal number of ASCII code for 0)
30 IF ASC (A$) = $30
      THEN GOTO 100  ← Jump to line number 100
```

3.3 CHR$ (Character)

Function: To give a character corresponding to the ASCII code representing the value of the expression.

Format: CHR$ (〈 Expression 〉)

Example:
```
10 FOR I = 48 TO 57
20 PRINT CHR$ ( I );      ← Display 0123456789
30 NEXT I
```

3.4 FRE (Free)

Function: To give the size of the memory area in units of bytes.

Format: FRE (⟨ Character ⟩)

Specify any value for the expression since ⟨ character ⟩ does not have any meaning.

Example: For specifying the memory space:

```
PRINT FRE ( 0 )
```

3.5 FLD (Field)

Function: To specify control of the field of the characters to be displayed in the inverted character mode and in the user's definition character mode.

Format: FLD ({ 0 | 1 })

Inverted mode	0: Initialize the normal field
	1: Initialize the inverted character field
User's definition mode	0: Initialize the normal field
	1: Initialize the user's definition field

The field can be specified for the display elements and print elements of the PRINT sentences and LPRINT sentences.

Example: In the inverted character mode:

```
PRINT "1234" ; FLD (1) ; "1234"
```

1234 ■ ■ ■ ■ is displayed.

Note: The field returns to the normal field after execution of the PRINT sentence.

3.6 HEX$

Function: To convert the value of the expression into a numeral of 2 or 4 digits in hexadecimal notation and to give its character-string.

Format: HEX$ (⟨ Expression ⟩)

Example:
```
10 I = 100
20 PRINT HEX$ ( I )       ← Display 64
```

3.7 HPOS, VPOS (Horizontal/Vertical Position)

Function: To output a horizontal position (HPOS) and a vertical position (VPOS) of the cursor.

Format: HPOS (〈Expression〉)

VPOS (〈Expression〉)

Specify any value for the expression since it does not have any meaning.

Example:
```
10 PRINT "♥",
20 A = VPOS (0)
30 B = HPOS (0)
40 LOCATE A, B
50 PRINT "◆"          ← Display ◆ at 8th digit in the same line of ♥
```

Note: HPOS and VPOS provide the position of the cursor immediately after execution of the preceding PRINT sentence.

3.8 LEFT$

Function: To extract and give the value of the character-string of the expression from the start of the character-string data stored in the character-string variable.

Format: LEFT$ (〈Character-String Variable〉, 〈Expression〉)

Example: $1 \leqq$ Expression $\leqq 32$

```
10 C$ = "ABC␣DEF"
20 PRINT LEFT$ (C$, 5)        ← Display ABC␣D
```

3.9 LEN (Length)

Function: To give the number of characters stored in the character-string variable.

Format: LEN (〈Character-String Variable〉)

Example:
```
10 D$ = "JR-100U␣BASIC"
20 PRINT LEN (D$)             ← Display 13
```

3.10 MID$

Function: To give character-string data stored in the character-string variable, a desired position of which in extracted.

Format: MID$ (〈Character-String Variable〉, 〈Expression 1〉, 〈Expression 2〉)

Expression 1: Position of the character to be extracted $1 \leqq$ 〈Expression〉$\leqq 32$

Expression 2: The number of characters to be extracted. In the case of omission, the characters are extracted up to the end.

Example:
```
10 D$ = "JR-100U␣␣BASIC"
20 PRINT MID$ (D$, 4, 3)      ← Display 100
```

3.11 MOD (Modula)

Function: To give the remainder of a division operation.

Format: MOD (〈Expression 1〉, 〈Expression 2〉)

Example: For obtaining the remainder of division of (A + B) by 2:

```
MOD (A + B, 2)              ← Residual when (A + B) is divided by 2.
```

For making a jump to a column address 100, if the value of A is an odd number:

```
50 IF MOD (A, 2) = 1 THEN GOTO 100
```

3.12 PEEK

Function: To provide data of 1 byte which is read out from the specified memory address.

Format: PEEK (〈Expression〉)

Example: For obtaining a total sum of data from address $1000 to address $100F of the memory.

```
10 S = 0
20 FOR I = $1000 TO $100F
30 S = S + PEEK (I)          ← Obtain total sum
40 NEXT I
```

3.13 RIGHT$

Function: To extract the number of the characters from the ~~left~~ right end of the character-string data stored in the character-string variable.

Format: RIGHT$ (〈Character-String Variable〉, 〈Expression〉)

where $1 \leqq$ 〈Expression〉$\leqq 32$

An error occurs when the value of the expression is 0.

Example:

```
10 C$ = "JR-100U BASIC"
20 PRINT RIGHT$ (C$, 5)     ← Display BASIC
```

3.14 RND (Random)

Function: To generate random numbers within the range of 0 to (〈Expression〉 −1)

Format: RND (〈Expression〉)

where $1 \leqq$ 〈Expression〉$\leqq 32767$

Example: For generating random numbers form 0 to 9:

```
RND (10)
```

For generating random numbers from 10 to 29:

```
RND (20) + 10
```

For randomly generating letters:

```
CHR$ (RND (26) + $41)
```

3.15 SGN (Sign)

Function: To examine the sign of the value and give a number representing it.

Format: SGN (⟨ Expression ⟩)

When ⟨ Expression ⟩ > 0, 1 is given.

When ⟨ Expression ⟩ = 0, 0 is given.

When ⟨ Expression ⟩ < 0, −1 is given.

Example: Jump to line numbers 300, 200 and 100, if A is positive, zero and negative, respectively.

```
GOTO (SGN (A) + 2) * 100
```

3.16 SPC (Space)

Function: To display and print the specified number of spaces.

Format: SPC (⟨ Expression ⟩)

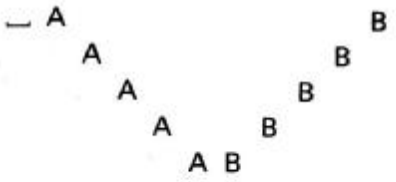
where 0 ≦ ⟨ Expression ⟩ ≦ 255

The spaces are used only for the display elements and print elements.

Example:

```
10 FOR I = 1 TO 5
20 PRINT SPC ( I ) ; "A" ; SPC ( (5 − I) * 2) ; "B"
30 NEXT I
```

⟨ Display ⟩



3.17 TAB (Tabulation)

Function: To move the cursor to the position specified by the expression for tabulating to the display (print) position.

Format: TAB (⟨ Expression ⟩)

where 0 ≦ ⟨ Expression ⟩ ≦ 255

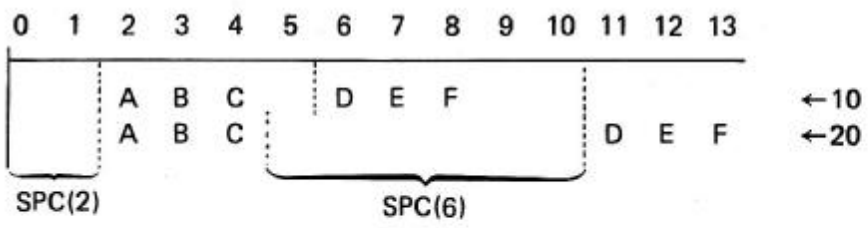
The tabulations are used only for the display elements and print elements.

Example:

```
10 PRINT TAB (2) ; "ABC" ; TAB (6) ; "DEF"
20 PRINT SPC (2) ; "ABC" ; SPC (6) ; "DEF"
```

The display on the screen is as shown below. Note the difference between TAB and SPC.

⟨ Screen ⟩



Note: If the value of the expression is less than the current display (print) position, the TAB command is neglected.

3.18 USR (User)

Function: To set a value in the register of the computer. The machine language program is accessed as a subroutine and a return is then made to the original program.

Format: USR (⟨ Expression 1 ⟩ [, ⟨ Expression 2 ⟩ [, ⟨ Expression 3 ⟩]])

Value of ⟨ Expression 1 ⟩ : set in the program counter

Value of ⟨ Expression 2 ⟩ : set in the B and A register

Value of ⟨ Expression 3 ⟩ : set in the B and A register

If ⟨ Expression 2 ⟩ and ⟨ Expression 3 ⟩ are omitted, 0 is set.

Example: The machine language program at address $1000 of the memory is accessed as a subroutine. 0 is set in 64 ($40) is set in the A register.

```
G = USR ($1000, 0, $0140)
```

The value of the B and A registers when returning from the machine language program are supplied to G after execution.

3.19 VAL (Value)

Function: To convert the character-string into a value.

Format: VAL (⟨ Character-String Variable ⟩)

Example:

```
10 A$ = "12"
20 B$ = "34"
30 C$ = A$ + B$
40 C = VAL (A$) + VAL (B$)
50 PRINT C$          ← 1234 is obtained.
60 PRINT C           ← 46 is obtained.
```

Note: Although the character-string is not a value, it can be converted into a value by the VAL function and can then be operated. If the character-string does not have a corresponding number, an error occurs.

No.	Error Message	Meaning
1	CANT CONTINUE	The program cannot be continued in the CONT command. The CONT command is only effective when the program is interrupted by the STOP sentence.
2	SUM CHECK ERROR	A LOAD error occurs. Load the program again at a different volume level.
3	DIVISION BY ZERO ERROR	The content of 〈expression〉 is 0 in the RND (〈Expression〉), MOD (N, 〈Expression〉). Alternatively, division by 0 is called for by the program.
4	FILE EMPTY ERROR	The SAVE command is input when no program is stored in the memory. An improper address is specified in the MSAVE command.
5	ILLEGAL LINE NO. ERROR	An address to be jumped to is absent in the GOTO command. An improper line number is selected in line editing on the screen.
6	INPUT ERROR	The INPUT sentence is used in the direct mode. 73 characters or more are keyed in at the keyboard.
7	LINE TOO LONG ERROR	A line is too long to be edited on the screen.
8	NESTING ERROR	The value variable of the "FOR" and the variable of "NEXT" do not coincide. Deep nesting level.
9	OUT OF DATA ERROR	Data which to be input by the READ command is not defined in the DATA command.
10	OUT OF MEMORY ERROR	The program is too long. The memory area for array, character-string is insufficient. The stacked area is insufficient because an expression is too complicated.
11	OVERFLOW ERROR	The input value and the calculated results are too large or too small.

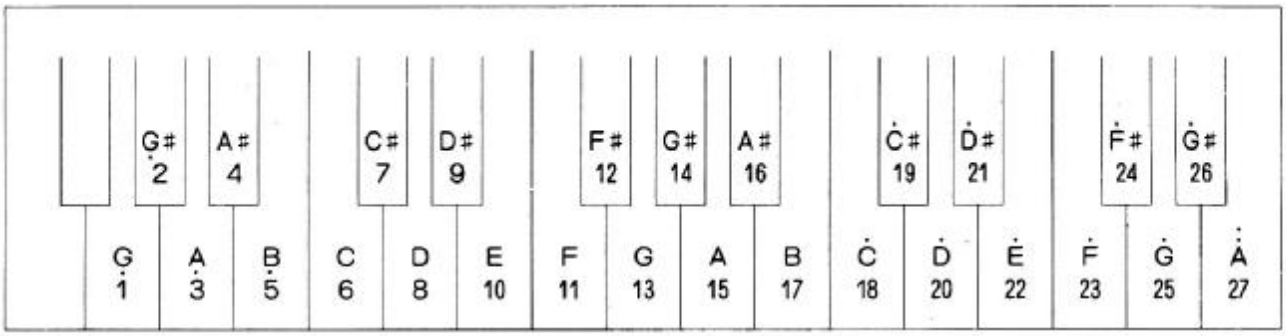
12	REDIM'D ARRAY ERROR	The same array name is used for two definitions.
13	SUBROUTINE CALL ERROR	The RET sentence does not correspond the GOSUB command. The subroutine nesting exceeds 15.
14	SUBSCRIPT ERROR	An addition to an array variable is out of the range of definition.
15	SYNTAX ERROR	The format is improper. Improper data is used.
16	TYPE MISMATCH ERROR	Different types of variables are mixed in. The specified data is of different types.
17	UNDEF'D ARRAY ERROR	An array which does not confirm to the DIM sentence is read out or written.
18	VALUE ERROR	Parameters which are specified, are out of the range of the character array function or other functions. A location which is not on the screen is specified with the LOCATE command.
19	VERIFY ERROR	The contents of a SAVED program are different from those in the memory.

(Appendix) MUSIC PROGRAM

JR-100U does not have a function to directly initiate a musical performance. However, utilizing the following program, you can enjoy the musical performance.

```
100 DIM A(27,2):POKE $C80B,$E0
110 RESTORE 910:FOR I=1 TO 27:READ A(I,2),A(I,1):NEXT I
120 FOR I=1 TO 10000
130 READ B,C
140 IF B=99 THEN 180
150 IF B=0 THEN GOSUB 190:NEXT I
160 POKE $C804,A(B,1):POKE $C805,A(B,2):GOSUB 200
170 NEXT I
180 POKE $C80B,0:END
190 POKE $C80B,0:GOSUB 200:POKE $C80B,$E0:RET
200 FOR W=1 TO 180*C:NEXT W:RET
900 REM FREQUENCY DATA
910 DATA 4,$73,4,$33,3,$F6,3,$BD,3,$87,3,$55,3,$25,2,$F7,2,$CD
920 DATA 2,$A4,2,$7E,2,$5A,2,$38,2,$18,1,$FA,1,$DD,1,$C2,1,$A9
930 DATA 1,$91,1,$7A,1,$65,1,$51,1,$3E,1,$2D,1,$1B,1,$0B,0,$FC
1000 DATA 13,2,13,2,20,2,20,2,22,2,22,2,20,4,18,2,18,2
1010 DATA 17,2,17,2,15,2,15,2,13,4,99,99
```

When the above program is executed, tones from the middle C to one-octave high C are produced. Let's explain a way to perform a melody with JR-100U. In the above program, a tone is produced by a set of pieces of data for the pitch and the duration (including a rest note). Use the following keyboard to determine the pitch of the note. The numbers written on the keys indicate the sound data. Data for a rest note is 0. The shortest note (including a rest note) is 1. (For example, when an eighth note is 1, a quarter note is 2 and a half note is 4).



Assume that all notes 𝅘𝅥𝅯 𝅘𝅥𝅯 𝅘𝅥 have the same pitch C. Note data is then given as 0, 1, 6, 1, 6, 1, 6, 1 , 6, 2.

Let's create a melody by note data. All the data sentences must be stored as the DATA sentences starting from address 1000. End up the melody with data 99. Read this data to terminate the program.

The tempo of the melody may vary in accordance with changes of 180 of 180*C in line number 200 in the list described above. Increase the number to increase the tempo and decrease the number to decrease it.

The length of the song cannot be limited.

The following program is created to repeat the first four measures of the song "Twinkle Twinkle Little Star". In order to repeat the melody, the sentence with line number 140 is modified as follows. Let's create other songs!

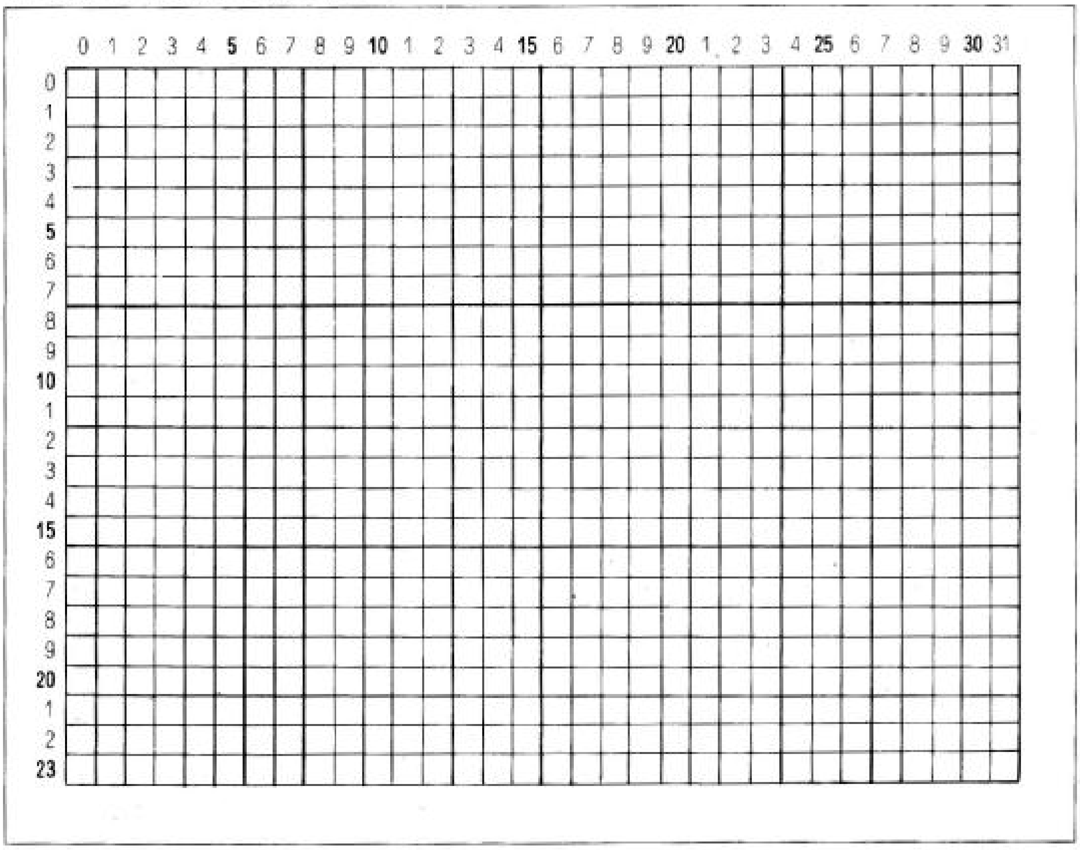
```
100 DIM A(27,2):POKE $C80B,$E0
110 RESTORE 910:FOR I=1 TO 27:READ A(I,2),A(I,1):NEXT I
120 FOR I=1 TO 10000
130 READ B,C
140 IF B=99 THEN RESTORE 1000:GOTO 120
150 IF B=0 THEN GOSUB 190:NEXT I
160 POKE $C804,A(B,1):POKE $C805,A(B,2):GOSUB 200
170 NEXT I
180 POKE $C80B,0:END
190 POKE $C80B,0:GOSUB 200:POKE $C80B,$E0:RET
200 FOR W=1 TO 120*C:NEXT W:RET
900 REM FREQUENCY DATA
910 DATA 4,$73,4,$33,3,$F6,3,$BD,3,$87,3,$55,3,$25,2,$F7,2,$CD
920 DATA 2,$A4,2,$7E,2,$5A,2,$38,2,$18,1,$FA,1,$DD,1,$C2,1,$A9
930 DATA 1,$91,1,$7A,1,$65,1,$51,1,$3E,1,$2D,1,$1B,1,$0B,0,$FC
1000 DATA 13,2,13,2,20,2,20,2,22,2,22,2,20,4,18,2,18,2
1010 DATA 17,2,17,2,15,2,15,2,13,4,99,99
```

(Appendix) ASCII Code Table

ASCII code		Char-	ASCII code		Char-	ACSII code		Char-	ASCII code		Char-	ASCII code		Char-
10 base	16 base	acter	10 base	16 base	acter	10 base	16 base	acter	10 base	16 base	acter	10 base	16 base	acter
000	00		052	34	4	104	68		156	9C	◣	208	D0	
001	01		053	35	5	105	69		157	9D	◱	209	D1	
002	02		054	36	6	106	6A		158	9E	⊠	210	D2	
003	03		055	37	7	107	6B		159	9F	◰	211	D3	
004	04		056	38	8	108	6C		160	A0		212	D4	
005	05		057	39	9	109	6D		161	A1		213	D5	
006	06		058	3A	:	110	6E		162	A2		214	D6	
007	07		059	3B	;	111	6F		163	A3		215	D7	
008	08		060	3C	<	112	70		164	A4		216	D8	
009	09		061	3D	=	113	71		165	A5		217	D9	
010	0A		062	3E	>	114	72		166	A6		218	DA	
011	0B		063	3F	?	115	73		167	A7		219	DB	
012	0C		064	40	@	116	74		168	A8		220	DC	
013	0D		065	41	A	117	75		169	A9		221	DD	
014	0E		066	42	B	118	76		170	AA		222	DE	
015	0F		067	43	C	119	77		171	AB		223	DF	
016	10		068	44	D	120	78		172	AC		224	E0	○
017	11		069	45	E	121	79		173	AD		225	E1	◪
018	12		070	46	F	122	7A		174	AE		226	E2	◪
019	13		071	47	G	123	7B		175	AF		227	E3	◪
020	14		072	48	H	124	7C		176	B0		228	E4	◪
021	15		073	49	I	125	7D		177	B1		229	E5	◧
022	16		074	4A	J	126	7E		178	B2		230	E6	◪
023	17		075	4B	K	127	7F		179	B3		231	E7	◪
024	18		076	4C	L	128	80		180	B4		232	E8	◨
025	19		077	4D	M	129	81	♠	181	B5		233	E9	⊞
026	1A		078	4E	N	130	82	♥	182	B6		234	EA	■
027	1B		079	4F	O	131	83	♦	183	B7		235	EB	⊞
028	1C		080	50	P	132	84	♣	184	B8		236	EC	⊞
029	1D		081	51	Q	133	85	♠	185	B9		237	ED	▩
030	1E		082	52	R	134	86	←	186	BA		238	EE	⬒
031	1F		083	53	S	135	87	↓	187	BB		239	EF	◳
032	20	(SP)	084	54	T	136	88	↑	188	BC		240	F0	◰
033	21	!	085	55	U	137	89	→	189	BD		241	F1	□
034	22	▾▾	086	56	V	138	8A	人	190	BE		242	F2	◧
035	23	#	087	57	W	139	8B	☺	191	BF		243	F3	◪
036	24	$	088	58	X	140	8C	■	192	C0		244	F4	◧
037	25	%	089	59	Y	141	8D	◱	193	C1		245	F5	◨
038	26	&	090	5A	Z	142	8E	■	194	C2		246	F6	◪
039	27	▾	091	5B	(	143	8F	◪	195	C3		247	F7	◧
040	28	(	092	5C	¥	144	90	◳	196	C4		248	F8	◰
041	29	)	093	5D	)	145	91	□	197	C5		249	F9	◨
042	2A	*	094	5E	^	146	92	□	198	C6		250	FA	◪
043	2B	+	095	5F	—	147	93	□	199	C7		251	FB	◰
044	2C	,	096	60		148	94	▬	200	C8		252	FC	●
045	2D	—	097	61		149	95	▬	201	C9		253	FD	◰
046	2E	.	098	62		150	96	■	202	CA		254	FE	◳
047	2F	/	099	63		151	97	■	203	CB		255	FF	◲
048	30	0	100	64		152	98	◳	204	CC				
049	31	1	101	65		153	99	⊞	205	CD				
050	32	2	102	66		154	9A	◫	206	CE				
051	33	3	103	67		155	9B	⊟	207	CF				

(Appendix) Design Chart

Location design chart (TV screen)



User's definition character

( )

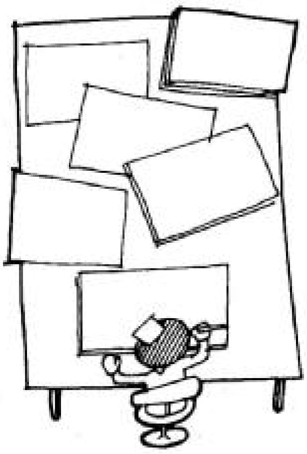
Address	7	6	5	4	3	2	1	0	Pattern code
( ) 0									( )
( ) 1									( )
( ) 2									( )
( ) 3									( )
( ) 4									( )
( ) 5									( )
( ) 6									( )
( ) 7									( )

( )

Address	7	6	5	4	3	2	1	0	Pattern code
( ) 0									( )
( ) 1									( )
( ) 2									( )
( ) 3									( )
( ) 4									( )
( ) 5									( )
( ) 6									( )
( ) 7									( )

( )

Address	7	6	5	4	3	2	1	0	Pattern code
( ) 0									( )
( ) 1									( )
( ) 2									( )
( ) 3									( )
( ) 4									( )
( ) 5									( )
( ) 6									( )
( ) 7									( )



Matsushita Electric Trading Co., Ltd.
Trade Center P.O.Box 18 Tokyo. 105.